On the Measure of Instability in Normal Residuated Logic Programs

Nicolás Madrid Manuel Ojeda-Aciego

Abstract— **Inconsistency in the framework of general residuated logic programs can be, somehow, decomposed in two notions: incoherence and instability. In this work, we focus on the measure of instability of normal residuated programs. Some measures were already provided and initial results obtained in terms of the amount of information that have to be discarded in order to recover stability; in this paper, our interest is focused precisely on the case in which stability can be recovered by adding information to our program.**

I. Introduction

In many fields of automated information processing it becomes crucial to consider together imprecise, uncertain or inconsistent information. Although inconsistency is an undesirable property, it arises naturally in many real-world problems (for instance, consider the integration of information coming from different sources). Anyway, the analysis of inconsistent knowledge-bases can lead us to obtain useful information: for instance, a big number of contradictions in the statements of a suspect of a crime with respect to the forensic evidences may lead us to increase our confidence on his/her being the culprit; a sensor which send data which contradict other sensors may indicate a possible malfunction. In both cases, a good estimation of the degree of inconsistency of the data can help us to estimate the truth-degree up to which this new information can be safely considered.

There are several papers dealing with inconsistency in a classical logic programming framework. For instance, [1] uses consistency restoring rules as a means to recover whenever possible the consistency of a normal logic program; this approach has been used in [2] to formalize negotiations dealing with incomplete information, preferences, and changing goals. The Answer Set Programming (ASP) framework has been used to detect inconsistencies in large biological networks [3]. Argumentation theory is a suitable framework for inconsistency to arise. There are several non-classical approaches to ASP argumentation, some based on possibility theory, some other based on, for instance, fuzzy set theory [4], [5].

The problem of measuring the degree of inconsistency contained in a knowledgebase has been already considered in the literature [6], [7], [8]. This approach shows that measuring the inconsistency of a knowledgebase is useful to allow for the comparison of the inconsistency of various knowledgebases. On the other hand, Lozinskii provided a method [9] for defining the quantity of information of a knowledgebase in propositional logic. However, that method is not suitable when the knowledgebase is inconsistent. Furthermore, it is certainly false that all inconsistent knowledgebases contain the same (null) amount of information, this is especially relevant when considering fuzzy extensions of the theory.

This work is based on the Fuzzy Answer Set Programming for residuated logic programs defined in [10], [11], in which we consider a fuzzy answer set attending to two dimensions: coherence and stability, the former is related to strong negation, whereas the latter is related to default negation and the GL-reduct [12]. An inconsistent fuzzy program is a program without fuzzy answer sets, and this can be due to the lack of stable models (instability) or, perhaps, to the inconsistency of every stable model (incoherence). This is why we talk about the two dimensions of inconsistency. In [13] some measures of inconsistency were defined in terms of incoherence; in this work, we aim at paving the way towards the measuring the degree of instability in normal residuated logic programs.

The structure of the paper is described as follows. In Section II we recall the definition of stable model. Section III describes the possible causes of the instability of a residuated logic program and defines the notion of information measure, which assigns a degree of information to any value in the truth space. In Section IV we define the measure of instability which establishes how many information has to be added on a set of rules in order to recovering the stability in the normal residuated logic program.

II. Preliminaries

Let us start this section 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 our 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 the rest of the paper we will consider a residuated lattice enriched with a negation operator, $(L, *, \leftarrow, \neg)$. The negation $\neg$ will model the notion of default negation often used in logic programming. As usual, a negation operator, over L,

N. Madrid is with the Dept. Matemática Aplicada, Univ. de Málaga, Spain. (email: nmadrid@ctima.uma.es).

M. Ojeda-Aciego is with the Dept. Matemática Aplicada, Univ. de Málaga, Spain. (email: aciego@ctima.uma.es).

is any decreasing mapping $n\colon L \to L$ satisfying $n(0) = 1$ and $n(1) = 0$. In the examples, we will use the following familiy of negation operators:

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

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

$$\langle p \leftarrow p_1 * \cdots * p_m * \neg p_{m+1} * \cdots * \neg p_n; \quad \vartheta \rangle$$

where ϑ is an element of L and $p, p_1, \ldots, p_n$ are propositional symbols.

It is usual to denote the rules as $\langle p \leftarrow \mathcal{B}; \vartheta \rangle$. The formula $\mathcal{B}$ is usually called the *body* of the rule whereas p 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 $p_1 * \cdots * p_m$ and the latter for $\neg p_{m+1} * \cdots * \neg p_n$.

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 propositional symbols appearing in $\mathbb{P}$ is denoted by $\Pi_{\mathbb{P}}$.

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

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

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 $I(p) \leq J(p)$ for all literal $p \in \Pi_{\mathbb{P}}$.*

A. Stable Models

Our aim in this section is to adapt the approach given in [12] to the normal residuated logic programs just defined in the section above.

Let us consider a normal residuated logic program $\mathbb{P}$ together with a fuzzy L-interpretation I. To begin with, we will construct a new normal program $\mathbb{P}_I$ by substituting each rule in $\mathbb{P}$ such as

$$\langle p \leftarrow p_1 * \cdots * p_m * \neg p_{m+1} * \cdots * \neg p_n; \quad \vartheta \rangle$$

by the rule[1]

$$\langle p \leftarrow p_1 * \cdots * p_m; \quad \neg I(p_{m+1}) * \cdots * \neg I(p_n) * \vartheta \rangle$$

Notice that the new program $\mathbb{P}_I$ is positive , that is, does not contain any negation; in fact, the construction closely resembles that of a reduct in the classical case, this is why we introduce the following:

[1]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 4: The program $\mathbb{P}_I$ is called the *reduct* of $\mathbb{P}$ wrt the interpretation I.

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 $\mathbb{P}_J$.

It is not difficult to prove that every model M of the program $\mathbb{P}$ is a model of the reduct $\mathbb{P}_M$.

Recall tha*t a fuzzy interpreta*tion can be interpreted as a L-fuzzy subset. Now, as usual, the notion of reduct allows for defining a *stable set* for a program.

Definition 5: Let $\mathbb{P}$ be a normal residuated logic program and let I be a fuzzy L-interpretation; I is said to be a *stable set* of $\mathbb{P}$ iff I is a minimal model of $\mathbb{P}_I$.

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

Thanks to Theorem 1 we know that every stable set is a model, therefore we will be able to use the term stable model to refer to a stable set. Obviously, this approach is a conservative extension of the classical approach.

In the following example we use a simple normal logic program with just one rule in order to clarify the definition of stable set (stable model).

Example 1: Consider the program $\langle p \leftarrow \neg q \quad ; \vartheta \rangle$. Given a fuzzy L-interpretation $I\colon \Pi \to L$, the reduct $\mathbb{P}_I$ is the rule (actually, the fact) $\langle p \quad ; \vartheta * \neg I(q) \rangle$ for which the least model is $M(p) = \vartheta * \neg I(q)$, and $M(q) = 0$. As a result, I is a stable model of $\mathbb{P}$ if and only if $I(p) = \vartheta * \neg I(0) = \vartheta * 1 = \vartheta$ and $I(q) = 0$. □

The following example shows that stable models for a normal residuated logic program need not exist.

Example 2: Consider the the following normal residuated logic program

$$\langle p \leftarrow n_\alpha(p) \quad ; 1 \rangle$$

defined over the residuated lattice $([0,1], \leq, *_P, \leftarrow_P, n_\alpha)$ (for any $\alpha \in [0, 1)$). This normal residuated logic program does not have stable models. Let I be an interpretation. The reduct w.r.t. I is either the fact $\langle p \leftarrow \quad ; 1 \rangle$ if $I(p) \leq \alpha$ or the fact $\langle p \leftarrow \quad ; 0 \rangle$ if $I(p) > \alpha$. In any case, if I is a stable model then $I(p)$ is equal either 1 or 0. However, both interpretations are not stable models of this normal residuated logic program. □

The aim of this work is to study normal residuated logic programs without any stable model by means of measures which determine how much information one has to add or delete in order to recover at least one stable model. We start by proposing the following definition:

Definition 6: A normal residuated logic program $\mathbb{P}$ is stable if and only if there is an L-interpretation such that $I = lfp(\mathbb{P}_I)$; i.e I is a stable model of $\mathbb{P}$. Otherwise, $\mathbb{P}$ is called unstable.

III. Causes of instability: measures of information

Instability is an undesirable feature of a logic program. When representing knowledge as a (residuated) logic program it is usual to implement rules according to a set of external data (obtained either from sensors or from suggestion of an expert); this data is subject to mistake and/or imprecisions, and may lead to the following shortcomings:

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

Any of the situations above might lead to instability. Let us further discuss this by means of an example: the following program tries to simulate a procedure to deduce which sports are practised by a person given some data.

$$\begin{aligned} r_1\colon \langle Football \leftarrow n_{0.4}(Basketball) *_G \\ *_G LivesInSuburb *_G AthleticBody \quad ; 0.6\rangle \\ r_2\colon \langle Basketball \leftarrow n_{0.4}(Cycling) *_G \\ *_G Tall *_G AthleticBody \quad ; 0.6\rangle \\ r_3\colon \langle Cycling \leftarrow n_{0.4}(Football) *_G \\ *_G Slim *_G AthleticBody \quad ; 0.6\rangle \end{aligned}$$

The first rule determines that if a person with an athletic body, which lives in a suburb and we do not know whether he practices regularly basketball, the this person practices football frequently (the interpretation of the other two rules is similar). These three rules do not imply any contradiction, in fact, the program consisting of the three rules has just one stable model $I_\perp$. However, if we add the following facts

$$\begin{aligned} r_4\colon \langle AthleticBody \leftarrow \quad ; 0.8\rangle \\ r_5\colon \langle LivesInSuburb \leftarrow \quad ; 1\rangle \\ r_6\colon \langle Tall \leftarrow \quad ; 0.7\rangle \\ r_7\colon \langle Slim \leftarrow \quad ; 0.8\rangle \end{aligned}$$

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

As we said 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 are changed to 0.39, therefore reducing the amount of information provided by those rules, the program would remain stable.

The approach above has already been considered in [14]; however, lack of information is more difficult to handle, since we do not know which rules are missing. In principle, there are three possibilities on 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 $\langle Football \leftarrow \quad ; 0.5\rangle$, the program gets stable again.
- Adding proper rules. In this case, the new rules permit to draw consequences which allow for recovering stability. For example, if we include the rule

  $$\langle Basketball \leftarrow AthleticBody *_G Slim *_G Tall; 0.5\rangle$$

  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 the data used in obtaining rules r_1, r_2 and r_3 were obtained in an upper-middle class neighbourhood in which an athletic body can be due to the practice of sports; however, rules $r_4, \ldots, r_7$ were based on lower-middle class neighbourhoods, in which an athletic body might be consequence from hard work, and not from the practice of sports. As a result, the missing information in the previous example might be due to not considering a new propositional symbol indicating the amount of physical work required by the job of the person we are talking about. Notice that by adding the literal $n_{0.4}(PhysicalWork)$ in the bodies of rules r_1, r_2 and r_3, and the fact $\langle PhysicalWork \leftarrow \quad ; 0.6\rangle$, the program gets stable again.

Our approach to this problem can be divided into two frameworks, based on measuring the instability of a program by means of the minimum amount of information which we have either to remove or to add in order to obtain a stable 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 removed or added.

We propose to fix an operator $m\colon L \to \mathbb{R}^+$ such that:

- $m(x) = 0$ if and only if $x = \perp$
- m is monotonic

such an operator will be called an *information measure*.

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

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

□

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

$$m(x) = \max\{n\colon \perp < x_1 < \cdots < x_n = x\}$$

In fact, it is an information measure: if $x \neq \perp$, then $\perp < x$, and this implies $m(x) \geq 1$. On the other hand, if $x < y$, then for all chain $\perp < x_1 < \cdots < x_n = x$ we have the chain

$\bot < x_1 < \cdots < x_n = x < x_{n+1} = y$ which has a greater length, and this implies $m(x) < m(y)$. □

Information measures will be used to determine the amount of information inherently contained in any element of the lattice. From now on, we will consider that any lattice has an associated information measure.

IV. Measuring instability of normal residuated logic programs by adding information

In this section we define an instability measure based on the amount of information that has to be added to an unstable program so that it gets stable. Contrariwise to the classical case, in which the only form to add information is by including new rules, in our framework we can as well increase their weights by some amount. A specific operator will be defined for this task. The approach used here can be seen as a dualization of that given in [14].

We start by fixing a t-conorm s to handle the values of $\mathcal{L}$ (recall that a t-conorm is a commutative and monotonic map $L \times L \to L$ satisfying $s(\top, x) = \top$ and $s(\bot, x) = x$). Fixed such a t-conorm, we can define the following operator to modify the weights of rules.

Given a normal residuated logic program $\mathbb{P}$, a set $\{\langle r_i; \vartheta_i\rangle\}_i$ of rules in $\mathbb{P}$ and a set of values $\{\varphi_i\}_i$ we define a new normal residuated logic program $\overline{O_{\mathbb{P}}}(\{\langle r_i; \vartheta_i\rangle\}_i, \{\varphi\}_i)$ as follows:

$$\overline{O_{\mathbb{P}}}(\{\langle r_i; \vartheta_i\rangle\}_i, \{\varphi\}_i) = (\mathbb{P} \smallsetminus \{\langle r_i; \vartheta_i\rangle\}_i) \cup \{\langle r_i; s(\vartheta_i, \varphi_i)\rangle\}_i$$

In other words, the operator $\overline{O_{\mathbb{P}}}$ changes the weights of $\langle r_j; \vartheta_j\rangle \in \{\langle r_i; \vartheta_i\rangle\}_i$ by the new value $s(\vartheta_j, \varphi_j)$.

Notice that the operator $\overline{O_{\mathbb{P}}}$ really increases the weights of the rules $\{\langle r_i; \vartheta_i\rangle\}_i$ in the program $\mathbb{P}$. Specifically, the higher the values φ_i, the higher the new weights of the rules.

The example below clarifies the behaviour of this operator.

Example 5: On the residuated lattice with negation $([0,1], \leq, *_P, \leftarrow_P, n)$, consider the following program:

$$r_1\colon \langle p \leftarrow_P q *_P t *_P n(t) \quad ; 0.7\rangle$$
$$r_2\colon \langle p \leftarrow_P t *_P n(s) \quad ; 0.8\rangle$$
$$r_3\colon \langle q \leftarrow_P n(v) \quad ; 0.2\rangle$$
$$r_4\colon \langle t \leftarrow_P s *_P u *_P n(v) \quad ; 0.9\rangle$$

Consider the t-conorm associated to the operator $\overline{O_{\mathbb{P}}}$ to be $s(x,y) = \min\{x + y, 1\}$. Then, the modified program $\overline{O_{\mathbb{P}}}(\{r_1, r_3\}, \{0.4, 0.7\})$ is the following:

$$r_1\colon \langle p \leftarrow_P q *_P t *_P \mathcal{N}(t) \quad ; 1\rangle$$
$$r_2\colon \langle p \leftarrow_P t *_P \mathcal{N}(s) \quad ; 0.8\rangle$$
$$r_3\colon \langle q \leftarrow_P \mathcal{N}(v) \quad ; 0.9\rangle$$
$$r_4\colon \langle t \leftarrow_P s *_P u *_P \mathcal{N}(v) \quad ; 0.9\rangle$$

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

However, 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 the previous section that instability might be due to missing facts or rules. The following pathological example exhibits this behaviour.

Example 6: Consider the underlying residuated lattice with negation $([0,1], \leq, *_G, \leftarrow_G, n_{0.5})$ and the program with just one rule

$$\mathbb{P} \equiv \langle p \leftarrow n_{0.5}(p) \quad ; 0.7\rangle$$

Then, it is not possible to recover stability by simply increasing the weight of its only rule. □

Measuring the minimal amount of information needed to recover stability we need to consider the 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 standpoint. In the following, we will see how this process can be 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 avoided.

Firstly, let us assume that when introducing occurrences of a new propositional symbol q, either positively or negatively, in the bodies of some rules $\{\langle p_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle\}_i \subseteq \mathbb{P}$ the resulting program is stable. That is, the program $\mathbb{P}^*$ obtained from $\mathbb{P}$ by substituting the rules $\langle p_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle$ by either $\langle p_i \leftarrow \mathcal{B}_i * q; \vartheta_i\rangle$ or $\langle p_i \leftarrow \mathcal{B}_i * \neg q; \vartheta_i\rangle$ 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 $\langle p_i \leftarrow \mathcal{B}_i; \vartheta_i\rangle$ by either $\langle p_i \leftarrow \mathcal{B}_i; \vartheta_i * M(q)\rangle$ or $\langle p_i \leftarrow \mathcal{B}_i *; \vartheta_i * n(M(q))\rangle$ (depending on whether q 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 q and, in this case we have just removed information from the program since $\vartheta_i \geq \vartheta_i * M(q)$ and $\vartheta_i \geq \vartheta_i * n(M(q))$. This is the case already studied in [14], and it needs not be considered here.

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

Since our aim here, is to measure the minimum amount of information required to recover stability, we will just need to take into account the possible addition of new facts not already included in the program.

Remark 1: 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 only aim of stabilizing the program, and not detecting the *really* missing information.

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

$$\overline{\mathbb{P}} = \mathbb{P} \cup \{\langle \ell_i \leftarrow ; \bot\rangle \colon \ell_i \in \Pi_{\mathbb{P}} \textit{ and } \langle \ell_i \leftarrow ; \vartheta\rangle \notin \mathbb{P}\}$$

This results in explicitly including facts for all the symbols occurring in the program and, this way, we will be able to add information to any of them by means of the operator $\overline{O_{\mathbb{P}}}$ defined above.

The more information needed to recover stability, the more unstable the program is. The amount of information included by $\overline{O_{\mathbb{P}}}(\{\langle r_i, \vartheta_i\rangle\}_i, \{\varphi_i\}_i)$ is obtained by means of an information measure m and the formula

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

Given a normal residuated logic program $\mathbb{P}$, we define the measure of instability of a set of rules $\{\langle r_i, \vartheta_i\rangle\}_i \subseteq \overline{\mathbb{P}}$ (w.r.t. $\mathbb{P}$) as follows:

$$\begin{aligned}\textsc{Instab}^{add}_{\mathbb{P}}(\{\langle r_i, \vartheta_i\rangle\}_i\}) = \\ = \inf\{\sum_{i \in \mathbb{I}} m(\varphi_i) \colon \overline{O_{\mathbb{P}}}(\{\langle r_i, \vartheta_i\rangle\}_i, \{\varphi_i\}_i) \textit{ is stable}\}\end{aligned}$$

Note that the definition provides a measure of 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 6 shows this situation, since $\textsc{Instab}^{add}_{\mathbb{P}}(\{r_1\})$ is undefined.

Example 7: On the residuated lattice with negation $([0,1], \leq, \wedge_P, \leftarrow_P, n_{0.4})$, let us consider the following unstable logic program:[2]

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

It is not difficult to check that this program does not have stable models. We will use the t-conorm $s(x,y) = \min\{x+y, 1\}$ and the Euclidean norm in the formulas above to measure the instability of the rules of the program. The first step to measuring instability is to consider the following facts:

$$\begin{aligned} r_7 &\colon \langle p \leftarrow \quad ; 0\rangle \qquad & r_8 &\colon \langle q \leftarrow \quad ; 0\rangle \\ r_9 &\colon \langle r \leftarrow \quad ; 0\rangle \qquad & r_{10} &\colon \langle t \leftarrow \quad ; 0\rangle \\ r_{11} &\colon \langle u \leftarrow \quad ; 0\rangle \qquad & r_{12} &\colon \langle v \leftarrow \quad ; 0\rangle \end{aligned}$$

[2]To increase readability, the subscripts P have been removed.

Note that it is not possible to restore stability of $\mathbb{P}$ by adding information to rules $r_1, r_2, r_3, r_4, r_5, r_6, r_{10}$ and r_{12} since if we changed their weights by 1, the program keeps being unstable. For the case of r_7, one can see that if its weight would be a value $\alpha > 0.4$, then the program would have a stable model; specifically,

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

However, if the weight of r_7 would be a value $\alpha \leq 0.4$, the program keeps unstable. Therefore

$$\textsc{Instab}^{add}_{\mathbb{P}}(\{r_7\}) = \inf\{||x|| \colon x > 0.4\} = 0.4$$

The measure of instability for rules r_8, r_9 and r_{11} can be computed similarly to the previous case of r_7, the results are shown below:

x	r_8	r_9	r_{11}
$\textsc{Instab}^{add}_{\mathbb{P}}(\{x\})$	0.4	0.4	0.4

Notice that the positive values of $\textsc{Instab}^{add}_{\mathbb{P}}$ obtained for rules r_7, r_8, r_9 and r_{11} indicate the possible existence of at least one missing rule which head is either p or q or r or u. □

Some results about $\textsc{Instab}^{add}_{\mathbb{P}}$ are given below:

Proposition 1: Let $\mathbb{P}$ be a normal residuated logic program.

- If $\mathbb{P}$ is stable then $\textsc{Instab}^{add}_{\mathbb{P}}(\{\langle r_i, \vartheta_i\rangle\}_i\}) = 0$ for all set of rules $\{\langle r_i, \vartheta_i\rangle\}_i\} \subseteq \mathbb{P}$.
- If $\textsc{Instab}^{add}_{\mathbb{P}}(\mathbb{P}) = 0$ then for all $\varepsilon > 0$ there exists a set $\{\varphi_i\}$ of values in L such that $\overline{O_{\mathbb{P}}}(\{\langle r_i, \vartheta_i\rangle\}_i, \{\varphi_i\}_i)$ is stable and $\sum_{i \in \mathbb{I}} m(\varphi_i) < \varepsilon$

The proposition above states that, although stability is not equivalent to having null instability measure, the former implies the latter and, whenever the program has null instability measure it is possible to recover stability by adding an amount of information below any prescribed bound.

An interested case occurs when the underlying lattice of truth-values is finite, since in this case stability and null instability measure coincide.

Corollary 1: Let $\mathcal{L} \equiv (L, \leq, \leftarrow, \wedge, \neg)$ be a residuated lattice such that L is finite. Then:

$$\mathbb{P} \text{ is stable if and only if } \textsc{Instab}^{add}_{\mathbb{P}}(\mathbb{P}) = 0$$

Finally, the following result establishes that $\textsc{Instab}^{add}_{\mathbb{P}}$ is antitonic with respect to the order between residuated logic programs.

Proposition 2: Let $\mathbb{P}$ be a normal residuated logic program and let $\{\langle r_i; \vartheta_i\rangle\} \subseteq \{\langle \overline{r_i}; \overline{\vartheta_i}\rangle\}$ be two sets of rules of $\mathbb{P}$ such that $\textsc{Instab}^{add}_{\mathbb{P}}$ is defined for both. Then:

$$\textsc{Instab}^{add}_{\mathbb{P}}(\{\langle r_i; \vartheta_i\rangle\}) \geq \textsc{Instab}^{add}_{\mathbb{P}}(\{\langle \overline{r_i}; \overline{\vartheta_i}\rangle\})$$

For the case of finite programs the instability measure of the full program is always defined.

Proposition 3: Let $\mathbb{P}$ be a finite normal residuated logic program. Then $\textsc{Instab}^{add}_{\mathbb{P}}(\overline{\mathbb{P}})$ is defined.

In the next section, we show an approach to measuring instability in terms of the computation of stable models of a modified version of the original program.

V. Computing the measures of instability

The aim of this section is to show that computing the value of $\text{INSTAB}^{add}_{\mathbb{P}}(\{\langle r_i;\vartheta_i\rangle\})$ is equivalent to computing the set of stable modelS of a specific logic program. For ease, we suppose $[0,1]$ is set of truth values and the set of rules $\{\langle r_i;\vartheta_i\rangle\}$ is a singleton.

To compute the measure of instability $\text{INSTAB}^{add}_{\mathbb{P}}$ we have to obtain what values $\lambda \in [0,1]$ satisfy that $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle,\lambda)$ is stable; we recall that $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle,\lambda)$ coincides with $\mathbb{P}$ except in the rule $\langle r_i;\vartheta_i\rangle$, which is changed by $\langle r_i; s(\vartheta_i,\lambda)\rangle$. How can we introduce the parameter λ in $\overline{\mathbb{P}}$ through propositional symbols? Let α and β be two propositional symbols not occurring in $\overline{\mathbb{P}}$. Consider the following set of rules:

$$\langle \alpha \leftarrow n(\beta) \quad ;1\rangle \tag{1}$$

$$\langle \beta \leftarrow n(\alpha) \quad ;1\rangle \tag{2}$$

where $n(x) = 1-x$. The set of stable models of this pair of rules is the set $\{M_\lambda \equiv (\alpha,\lambda);(\beta,\lambda)\}_{\lambda\in[0,1]}$. Notice that for any $\lambda \in [0,1]$ there is a stable model M_λ such that $M_\lambda(\alpha) = \lambda$. Given $\mathbb{P}$, consider a new residuated logic program $\overline{\mathbb{P}}^\star$ by considering $\overline{\mathbb{P}} \setminus \{r_i\}$, together with rules (1) and (2), and the rules:

$$r_i^\star\colon \langle p_i \leftarrow \mathcal{B} * n(\gamma) \quad ;1\rangle$$

$$r_i^{\star\star}\colon \langle \gamma \leftarrow t_s(n(\vartheta), n(\alpha)) \quad ;1\rangle$$

where γ is a propositional symbol which not occurring in $\overline{\mathbb{P}}$ and t_s is the t-norm $t_s(x,y) = n(s(n(x),n(y)))$. Then the following results hold:

Lemma 1: Let $\mathbb{P}$ be a normal residuated logic program. Let M be an interpretation $M\colon \Pi_{\mathbb{P}} \cup \{\alpha,\beta,\gamma\} \to [0,1]$ such that:

$$M(\beta) = 1 - M(\alpha) \qquad M(\gamma) = t_s(1-\vartheta, 1-M(\alpha))$$

Then, we have that

1) If N is a model of $\overline{\mathbb{P}}^\star_M$, then it is also a model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$.
2) Reciprocally, any model N of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$ can be extended to a model of $\overline{\mathbb{P}}^\star_M$.

Proof: Assume that N is a model of $(\overline{\mathbb{P}}^\star)_M$. For each rule r_j of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$ different from $\langle r_i; s(\vartheta_i, M(\alpha))\rangle_M$ the proof is trivial since r_j belongs to $(\overline{\mathbb{P}}^\star)_M$ as well; thus, we only need to consider the case of the rule $\langle r_i; s(\vartheta_i, M(\alpha))\rangle_M$. As N satisfies rule $(r_i^\star)_M$:

$$N(p_i) \geq N(\mathcal{B}^+) * M(\mathcal{B}^-) * n(M(\gamma))$$

Therefore,

$$\begin{aligned} N(p_i) &\geq N(\mathcal{B}^+) * M(\mathcal{B}^-) * n(t_s(1-\vartheta, 1-M(\alpha)))) \\ &= N(\mathcal{B}^+) * M(\mathcal{B}^-) * s(\vartheta, M(\alpha)) \end{aligned}$$

Reciprocally, assume now that N is a model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$. Then, for each rule r_j of $(\overline{\mathbb{P}}^\star)_M$ different of $(1)_M$, $(2)_M$, $(r_i^\star)_M$ and $(r_i^{\star\star})_M$ the proof is trivial since r_j belongs to $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_{M|_{\Pi_{\mathbb{P}}}}$ as well. For rules $(1)_M$, $(2)_M$, $(r_i^\star)_M$ and $(r_i^{\star\star})_M$ the interpretation N has to be extended, and the values of $N(\beta)$ and $N(\gamma)$ can be defined by using $M(\alpha)$ as follows:

$$N(\beta) = 1 - M(\alpha) \qquad N(\gamma) = t_s(1-\vartheta, 1-M(\alpha))$$

Thus the fulfillment of $(1)_M$, $(2)_M$ and $(r_i^{\star\star})_M$ is trivial. For the rule $(r_i^\star)_M$, as N satisfies the rule $\langle r_i; s(\vartheta_i,\lambda)\rangle_M$, the following inequality holds:

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

Therefore, using that $s(x,y) = n(t_s(1-x,1-y))$:

$$\begin{aligned} M(p_i) &\geq N(\mathcal{B}^+) * M(\mathcal{B}^-) * s(\vartheta_i,\lambda) \\ &= N(\mathcal{B}^+) * M(\mathcal{B}^-) * n(t_s(1-\vartheta, 1-M(\alpha)) \\ &= N(\mathcal{B}^+) * M(\mathcal{B}^-) * n(M(\gamma)) \end{aligned}$$

Thus, $(r_i^\star)_M$ is satisfied by N. ∎

Proposition 4: Let $\mathbb{P}$ be a normal residuated logic program.

1) If M is a stable model of $\overline{\mathbb{P}}^\star$, then it is also a stable model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))$.
2) Reciprocally, any stable model M of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))$ can be extended to a stable model of $\overline{\mathbb{P}}^\star$.

Proof: Let M be a stable model of $\overline{\mathbb{P}}^\star$. Then, M is the least model of $(\overline{\mathbb{P}}^\star)_M$. Then, as the only rule whose head is β in $(\overline{\mathbb{P}}^\star)_M$ is:

$$\langle \beta \leftarrow \quad ; n(\alpha)\rangle$$

The value $M(\beta)$ has to be necessarily $1-M(\alpha)$. Similarly, we can infer that $M(\gamma) = t_s(1-\vartheta, 1-M(\alpha))$. Hence, the hypothesis of Lemma 1 hold. Therefore, M is a model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$ as well. In fact, $M|_{\Pi_{\mathbb{P}}}$ is a stable model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))$ since if there were a model $N \subset M|_{\Pi_{\mathbb{P}}}$ of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_{M|_{\Pi_{\mathbb{P}}}}$ then, by Lemma 1, N can be extended a model of $(\overline{\mathbb{P}}^\star)_M$ satisfying $N \subset M$; which is a contradiction with M being a stable model of $\overline{\mathbb{P}}^\star$.

Let M be a stable model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))$. Then, M is the least model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$. Then, if we extend M to β and γ as $M(\beta) = 1-M(\alpha)$ and $M(\gamma) = t_s(1-\vartheta, 1-M(\alpha))$ respectively, we can apply Lemma 1, thus M is also a model of $(\overline{\mathbb{P}}^\star)_M$. Should M not be the least model of $(\overline{\mathbb{P}}^\star)_M$ then, there would be a model N of $(\overline{\mathbb{P}}^\star)_M$ such that $N \subset M$. Note that necessarily in this case $M(\beta) = N(\beta)$ and $M(\gamma) = N(\gamma)$ by using the same argument as above, therefore there exists a propositional symbol q in $\Pi_{\mathbb{P}}$ such that $N(q) < M(q)$. Using again Lemma 1, $N \subset M$ is a model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))_M$; contradicting that M is a stable model of $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, M(\alpha))$

∎

Proposition 4 shows that there is an univocal correspondence among the stable model of $\overline{\mathbb{P}}^\star$ and the parameters λ_i such that $\overline{O_{\mathbb{P}}}(\langle r_i;\vartheta_i\rangle, \lambda_i)$ is stable. Therefore we can compute $\text{INSTAB}^{add}_{\mathbb{P}}(\{\langle r_i;\vartheta_i\rangle\})$ by using the stable models of $\overline{\mathbb{P}}^\star$:

Corollary 2: Let $\mathbb{P}$ be a normal residuated logic program. Then:

$$\text{INSTAB}^{add}_{\mathbb{P}}(\langle r_i; \vartheta_i \rangle) = \inf\{m(M(\alpha)) \quad | \ M \textit{ is a stable model of } \overline{\mathbb{P}}^{\star}\}$$

Note that by using monotonicity of m, we do not need to compute all stable model of $\overline{\mathbb{P}}^{\star}$, but only the stable model which assigns to the propositional symbol α the least truth value.

Example 8: The aim of this example is to show how applying the procedure above to a normal residuated logic program. Consider the negation operator:

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

And the normal residuated logic program $\mathbb{P}$ defined over the residuated lattice $([0,1], \leq, *_P, \leftarrow_P, \overline{n})$:

$$\begin{aligned} r_1 &: \langle p \leftarrow s * \neg q \quad ; 1.0 \rangle \\ r_2 &: \langle q \leftarrow \neg r \quad ; 0.9 \rangle \\ r_3 &: \langle r \leftarrow \neg p \quad ; 0.9 \rangle \\ r_4 &: \langle s \leftarrow \neg t * \neg u \quad ; 1.0 \rangle \end{aligned}$$

The reader can easily check that $\mathbb{P}$ is unstable. To apply $\text{INSTAB}^{add}_{\mathbb{P}}$ we will consider the information measure given by $m(x) = x^2$, the t-conorm $s(x,y) = \max\{x, y\}$ and the completion $\overline{\mathbb{P}}$ by adding the rules:

$$\begin{aligned} r_5 &: \langle p \leftarrow \quad ; 0 \rangle & r_6 &: \langle q \leftarrow \quad ; 0 \rangle \\ r_7 &: \langle r \leftarrow \quad ; 0 \rangle & r_8 &: \langle s \leftarrow \quad ; 0 \rangle \\ r_9 &: \langle t \leftarrow \quad ; 0 \rangle & r_{10} &: \langle u \leftarrow \quad ; 0 \rangle \end{aligned}$$

We start by computing the value of $\text{INSTAB}^{add}_{\mathbb{P}}(r_9)$. Thus, rule r_9 is removed from $\mathbb{P}$ and is substituted by the following four rules:

$$\begin{aligned} r_{\alpha_t} &: \langle \alpha_t \leftarrow n(\beta_t) \quad ; 1 \rangle \\ r_{\alpha_t} &: \langle \beta_t \leftarrow n(\alpha_t) \quad ; 1 \rangle \\ r_9^{\star} &: \langle t \leftarrow n(\gamma_t) \quad ; 1 \rangle \\ r_9^{\star\star} &: \langle \gamma_t \leftarrow n(\alpha_t) \quad ; 1 \rangle \end{aligned}$$

where $n(x) = 1 - x$. The reader can check that the family of stable models of this new normal residuated logic program is:

$$ST = \{\{(p, 1-\lambda); (q, 0); (r, 0.9); (t, \lambda); (u, 0); (s, 1-\lambda); (\alpha_t, \lambda)\} : \lambda > 0.5\}$$

Therefore:

$$\text{INSTAB}^{add}_{\mathbb{P}}(r_9) = \inf\{m(M(\alpha_t)) \mid M \in ST\} = \inf\{\lambda^2 \mid \lambda > 0.5\} = 0.25$$

□

VI. CONCLUSIONS

We have continued our study of fuzzy answer set semantics for residuated logic programs by focusing on the measure of instability of normal residuated programs. Some measures have been provided and initial results have been obtained, in terms of the amount of information that have to be added in order to recover stability.

As future work, we will study the dual situation in which stability can be recovered by adding information (as in the framework of consistency restoring rules). In addition, we will extend this methodology to provide explanations for inconsistencies in the data by determining minimal representations of conflicts. In practice, this can be used to identify unreliable data or to indicate missing reactions.

REFERENCES

[1] M. Balduccini and M. Gelfond, "Logic programs with consistency-restoring rules," in *Intl Symp on Logical Formalization of Commonsense Reasoning, AAAI 2003*, 2003, pp. 9–18.

[2] T. C. Son and C. Sakama, "Negotiation using logic programming with consistency restoring rules," in *IJCAI'09: Proc 21st Intl Joint Conf on Artificial intelligence*. Morgan Kaufmann Publishers Inc., 2009, pp. 930–935.

[3] M. Gebser, T. Schaub, S. Thiele, B. Usadel, and P. Veber, "Detecting inconsistencies in large biological networks with answer set programming," in *ICLP*, ser. Lecture Notes in Computer Science, vol. 5366, 2008, pp. 130–144.

[4] J. C. Nieves, U. Cortés, and M. Osorio, "Possibilistic-based argumentation: An answer set programming approach," *Mexican International Conference on Computer Science*, pp. 249–260, 2008.

[5] J. Janssen, M. De Cock, and D. Vermeir, "Fuzzy argumentation frameworks," in *Proc. of IPMU'08*, 2008, pp. 513–520.

[6] A. Hunter and S. Konieczny, "Approaches to measuring inconsistent information," in *Inconsistency Tolerance*, ser. Lecture Notes in Computer Science, vol. 3300, 2005, pp. 191–236.

[7] J. Grant and A. Hunter, "Measuring inconsistency in knowledgebases," *J. Intell. Inf. Syst.*, vol. 27, no. 2, pp. 159–184, 2006.

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

[9] E. Lozinskii, "Information and evidence in logic systems," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 6, pp. 163—193, 1994.

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

[11] ——, "Towards a fuzzy answer set semantics for residuated logic programs," in *Proc of WI-IAT'08. Workshop on Fuzzy Logic in the Web*, 2008, pp. 260–264.

[12] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proc. of ICLP-88*, 1988, pp. 1070–1080.

[13] 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)*, 2009, pp. 598–603.

[14] ——, "Measuring instability in normal residuated logic programs," 2010. Submitted.