On the use of fuzzy stable models for inconsistent classical logic programs

Nicolás Madrid and Manuel Ojeda-Aciego

***Abstract*—We introduce a sufficient condition which guarantees the existence of stable models for a normal residuated logic program interpreted on the truth-space $[0,1]^n$. Specifically, the continuity of the connectives involved in the program ensures the existence of stable models. Then, we focus on the assignment of a fuzzy stable model semantics to inconsistent classical logic programs on the basis of the separation of the notion of inconsistence and uncertainty.**

I. Introduction

We propose to apply a certain fuzzy semantics to interpret a crisp logic program, in some sense in the style of the well-founded semantics, which assigns one tri-valued stable model to a given crisp normal logic program.

Note, however, that the well-founded semantics does not provide a clear distinction between instability (absence of stable models) and uncertainty. The following example shows that feature.

$$\mathbb{P}_1 = \{p \leftarrow \neg p \ ; \ q \leftarrow \neg q\}$$
$$\mathbb{P}_2 = \{p \leftarrow \neg q \ ; \ q \leftarrow \neg p\}$$

have the same well-founded model $WF = \{(p,u),(q,u)\}$ (where u represents the value $unknown$). In the former case, the undefinedness is due to the explicit contradiction in the rules of the program whereas, in the latter case, u represents the uncertainty in knowing whether p is false and q true or vice versa.

In order to obtain a more expressive approach, one can consider the stable model semantics, which allows to represent the uncertainty regarding the value of the propositional symbols by considering several possibilities, one per existing stable model. However, in general, the existence of stable models cannot be guaranteed, and necessary conditions to ensure the existence of stable models have been widely studied in classical logic programming. In fact, a syntactic condition on crisp normal programs to have stable models can be found in [1]. Similarly to classical logic programming, the existence of fuzzy stable models cannot be guaranteed for an arbitrary normal residuated logic program [2].

Moreover the characterization in the fuzzy framework is much more complicated since it involves two different dimensions: the syntactic one "the structure of the normal program" and semantic one "the choice of suitable connectives in the residuated lattice".

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).

In classical logic programming only syntactic conditions are available since the connectives are fixed. However, for normal residuated logic program the semantic dimension plays also a crucial role; for example the program with only one rule

$$\mathbb{P} = \{\langle p \leftarrow \neg p; 1\rangle\}$$

has a stable model if and only if the operator associated with $\neg$ has a fixpoint. As far as we know, establishing semantic conditions for guaranteeing the existence of stable models has not been directly attempted, although sufficient conditions underlie in some approaches; for example [3] proves that every normal logic program has stable models in the 3-valued Kleene logic and, more generally, [4], [5] show that every normal residuated logic program has stable models if the underlying residuated lattice has an appropriate bilattice structure [6].

In this paper, firstly, we provide another condition on the residuated lattice to ensure the existence of stable models, more specifically: if the lattice selected is an Euclidean space and the connectives $*$ and $\neg$ in the residuated lattice are continuous, then the existence of at least a fuzzy stable model is guaranteed. Then, given a crisp logic program, we embed it into a residuated structure which guarantees the existence of stable models, and a preference among these stable models is introduced such that: (1) if the crisp program has stable models, the "residuated" semantics coincides with the classical one, and (2) otherwise, the residuated stable models allow to establish a clear distinction between instability and uncertainty.

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 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* $y*z \leq x \quad \forall x,y,z \in L$.

Remark 1. *The adjoint pair is uniquely determined by the chosen t-norm* $*$. *In other words, fixed a left-continuous t-norm* $*$, *the unique operator* $\leftarrow$ *which forms an adjoint pair together* $*$, *is that defined by:*

$$x \leftarrow y = \sup\{z \in L \colon y * z \leq x\}$$

This is the reason why we usually present residuated lattices by simply mentioning the operator $*$.

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, is any decreasing mapping $n\colon L \to L$ satisfying $n(0) = 1$ and $n(1) = 0$.

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 $p, p_1, \dots, p_n$ 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, p is called its *head* and ϑ is called its weight.

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

Note that the ordering 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 propositional symbol $p \in \Pi_{\mathbb{P}}$.*

A. Immediate Consequence Operator

The immediate consequence operator was successfully generalized for positive residuated programs in [7] and can be applied straightforwardly to normal residuated programs. Its definition is as follows:

Definition 4. *Let $\mathbb{P}$ be a normal residuated logic program. The immediate consequence operator maps every L-interpretation I to the L-interpretation $T_{\mathbb{P}}(I)$ defined below:*

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

where $p \in \mathbb{P}$.

Similarly to the positive case, the operator $T_{\mathbb{P}}$ can be used to characterize the models of normal residuated logic programs:

Proposition 1. *Let $\mathbb{P}$ be a residuated logic program and let M be an L-interpretation. M is a model of $\mathbb{P}$ if and only if $T_{\mathbb{P}}(M) \leq M$.*

Proof: Let M be a model of $\mathbb{P}$. Then for every rule $\langle p \leftarrow \mathcal{B}; \ \vartheta\rangle \in \mathbb{P}$:

$$M(p) \geq M(\mathcal{B}) * \vartheta$$

That implies that for every propositional symbol p, $M(p)$ is an upper bound of the set $\{M(\mathcal{B}) * \vartheta\colon \langle p \leftarrow \mathcal{B}; \ \vartheta\rangle \in \mathbb{P}\}$ and thus:

$$M(p) \geq \sup\{M(\mathcal{B}) * \vartheta\colon \langle p \leftarrow \mathcal{B}; \ \vartheta\rangle \in \mathbb{P}\} = T_{\mathbb{P}}(M)(p)$$

Assume now that $M(p) \geq T_{\mathbb{P}}(M)(p)$ for every propositional symbol p, then for every rule $\langle p \leftarrow \mathcal{B}; \ \vartheta\rangle$ in $\mathbb{P}$:

$$\begin{aligned} M(p) \geq T_{\mathbb{P}}(M)(p) = \sup\{M(\mathcal{B}') * \vartheta'\colon \ \langle p \leftarrow \mathcal{B}'; \ \vartheta'\rangle\} \geq \\ \geq M(\mathcal{B}) * \vartheta \end{aligned}$$

■

The immediate consequence operator is monotonic when is defined on positive residuated logic programs [7]:

Theorem 1. *Let $\mathbb{P}$ be a positive residuated logic program, then $T_{\mathbb{P}}$ is monotonic.*

The theorem above together with Knaster-Tarski's fix-point theorem ensure that the operator $T_{\mathbb{P}}$ has a least fix-point; furthermore this least fix point coincides with the least model of $\mathbb{P}$.

The main difference in the case of normal residuated logic programs, is that $T_{\mathbb{P}}$ is not necessarily monotonic. That feature implies that we cannot make use of the least model semantics in arbitrary normal residuated logic programs.

Example 1. *Consider the logic program $\langle p \leftarrow \neg q \quad ; 1\rangle$ interpreted on the residuated lattice with negation $([0, 1], \leq, \min, \leftarrow, 1 - x)$. Then the immediate consequence operator is the mapping:*

$$T_{\mathbb{P}}(I)(x) = \begin{cases} 1 - I(q) & \text{if } \ x = p \\ 0 & \text{otherwise} \end{cases}$$

where I is a $[0, 1]$-interpretation. Clearly this mapping is not monotonic with respect to the order between $[0, 1]$-interpretations.

Certainly, the definition of $T_{\mathbb{P}}$ can be simplified if for each propositional symbol p, there exists only one rule whose head is p, since the operator $\sup$ can be removed from the definition. Although that condition on a program $\mathbb{P}$ does not usually hold, we can always obtain a partition of $\mathbb{P}$ such that the condition holds for each partition and, then, the immediate consequence operator of $\mathbb{P}$ can be obtained by using the immediate consequence operator of each partition. Formally:

Proposition 2. *Let $\mathbb{P}$ be a normal residuated logic program. Then there exist a partition $\{\mathbb{P}_i\}_{i \in I}$ of the program $\mathbb{P}$ satisfying:*

- *For all $i \in I$, there are no rules in $\mathbb{P}_i$ with the same head.*
- *$T_{\mathbb{P}}(I)(p) = \sup_{i \in I}\{T_{\mathbb{P}_i}(I)(p)\}$.*

Proof: The finest partition of $\mathbb{P}$ satisfies the statement of the proposition. Explicitly, for each rule $r_i \in \mathbb{P}$ we consider the normal residuated logic program with just one rule $\mathbb{P}_i = \{r_i\}$.

Then the partition $\{\mathbb{P}_i\}_{i\in I}$ satisfies the first item. Now, for each $\mathbb{P}_i$ the immediate consequence operator has the form

$$T_{\mathbb{P}_i}(I)(x) = \begin{cases} I(\mathcal{B}) * \vartheta & \text{if } x = p \\ 0 & \text{otherwise} \end{cases}$$

where $\langle p \leftarrow \mathcal{B};\ \vartheta\rangle$ is the only rule in $\mathbb{P}_i$. Then:

$$T_{\mathbb{P}}(I)(p) = \sup\{I(\mathcal{B})*\vartheta : \langle p \leftarrow \mathcal{B};\ \vartheta\rangle \in \mathbb{P}\} = \sup_{i\in I}\{T_{\mathbb{P}_i}(I)(p)\}$$

■

B. Stable Models

We recall here the approach given in [8], which generalizes the stable model semantics [9] to normal residuated logic programs.

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:

Definition 5. *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 an L-fuzzy subset. Now, as usual, the notion of reduct allows for defining a *stable set* for a program.

Definition 6. *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 2. *Any stable set of $\mathbb{P}$ is a minimal model of $\mathbb{P}$.*

Thanks to Theorem 2 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).

[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.

Example 2. 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 $I(q) = 0$. □

An important feature of the stable models that holds as well in our extended framework is that a stable model is always a minimal fix-point of $T_{\mathbb{P}}$.

Proposition 3. *Any stable model of $\mathbb{P}$ is a minimal fix-point of $T_{\mathbb{P}}$.*

Proof: We will refer here a rule $\langle p \leftarrow p_1 * \cdots * p_m * \neg p_{m+1} * \cdots * \neg p_n;\quad \vartheta\rangle$ by writing $\langle p \leftarrow \mathcal{B}^+ * \mathcal{B}^-;\ \vartheta\rangle$ where $\mathcal{B}^+$ is identified with $p_1 * \cdots * p_m$ and $\mathcal{B}^-$ is identified with $\neg p_{m+1} * \cdots * \neg p_n$. With this notation, the reduct $\mathbb{P}_I$ can be seen as the transformation which substitutes each rule $\langle p \leftarrow \mathcal{B}^+ * \mathcal{B}^-;\ \vartheta\rangle$ in $\mathbb{P}$ by $\langle p \leftarrow \mathcal{B}^+;\ I(\mathcal{B}^-) * \vartheta\rangle$.

Let us see firstly that for every L-interpretation I, $T_{\mathbb{P}}(I) = T_{\mathbb{P}_I}(I)$:

$$\begin{aligned} &T_{\mathbb{P}}(I)(p) = \\ &= \sup\{I(\mathcal{B}^+) * I(\mathcal{B}^-) * \vartheta\colon \langle p \leftarrow \mathcal{B}^+ * \mathcal{B}^-;\ \vartheta\rangle \in \mathbb{P}\} = \\ &= \sup\{I(\mathcal{B}^+) * I(\mathcal{B}^-) * \vartheta\colon \langle p \leftarrow \mathcal{B}^+;\ I(\mathcal{B}^-) * \vartheta\rangle \in \mathbb{P}_I\} = \\ &= T_{\mathbb{P}_{\mathbb{I}}}(I)(p) \end{aligned}$$

Now, let M be a stable model of $\mathbb{P}$. Then, by definition, $M = T_{\mathbb{P}_M}(M)$. By using the equality above, we obtain $M = T_{\mathbb{P}_M}(M) = T_{\mathbb{P}}(M)$; in other words, M is a fix-point of $T_{\mathbb{P}}$.

Let us prove the minimality of M. Let N be a fix-point of $T_{\mathbb{P}}$ such that $N \leq M$, then by Proposition 1 N is a model of $\mathbb{P}$. Finally, by Theorem 2, $N = M$. ■

Notice, however, that a minimal fix-point of $T_{\mathbb{P}}$ is not necessarily a stable model of $\mathbb{P}$, as shown in the following example:

Example 3. *Let $\mathbb{P} = \{\langle p \leftarrow p; 1\rangle, \langle q \leftarrow \neg p; 1\rangle\}$ be a normal residuated logic program defined on $([0,1], \leq, \min, \leftarrow, 1-x)$. Let us obtain firstly the stable models of $\mathbb{P}$. Let $I = \{(p,\alpha),(q,\beta)\}$ be a $[0,1]$-interpretation, then the reduct $\mathbb{P}_I$ is the program $\mathbb{P}_I = \{\langle p \leftarrow p; 1\rangle, \langle q \leftarrow; 1-\alpha\rangle\}$. The least model of $\mathbb{P}_I$ is the $[0,1]$-interpretation $M = \{(p,0),(q,1-\alpha\}$. So I is a stable model of $\mathbb{P}$ if and only if $I = M$, that is, if and only if $I = \{(p,0),(q,1)\}$.*

Let us obtain now the set of fix-points of $T_{\mathbb{P}}$. The immediate consequence operator of $\mathbb{P}$ is:

$$T_{\mathbb{P}}(I)(x) = \begin{cases} I(p) & \textit{if} \quad x = p \\ 1 - I(p) & \textit{if} \quad x = q \end{cases}$$

A $[0,1]$-interpretation $I = \{(p,\alpha),(q,\beta)\}$ is a fix-point of $T_{\mathbb{P}}$ if and only if $I(p) = I(p)$ and $I(q) = 1 - I(p)$. Therefore the set of fix-points of $T_{\mathbb{P}}$ is $Fp = \{I_\alpha = \{(p,\alpha),(q,1-\alpha)\}\colon\ \alpha \in [0,1]\}$. Note that every $[0,1]$-interpretation in Fp is a minimal fix-point but only one of them is a stable model. □

III. On the existence of stable models in $[0,1]$

The existence of stable models can be guaranteed by simply imposing conditions on the underlying residuated lattice:

Theorem 3. *Let $\mathcal{L} \equiv ([0,1], \leq, *, \leftarrow, \neg)$ be a residuated lattice with negation. If $*$ and $\neg$ are continuous operators, then every finite normal program $\mathbb{P}$ defined over $\mathcal{L}$ has at least a stable model.*

Proof: The idea is to apply Brouwer's fix-point theorem. Specifically, we show that the operator assigning each interpretation I the interpretation $\mathcal{R}(I) = \mathrm{lfp}(T_{\mathbb{P}_I})$ is continuous. Note that this operator can be seen as a composition of two operators $\mathcal{F}_1(I) = \mathbb{P}_I$ and $\mathcal{F}_2(\mathbb{P}) = \mathrm{lfp}(T_{\mathbb{P}})$. Actually, we will show that $\mathcal{F}_1$ and $\mathcal{F}_2$ are continuous.

To begin with, note that $\mathcal{F}_1$ can be seen as an operator from the set of $[0,1]$-interpretations to the Euclidean space $[0,1]^k$ where k is the number of rules in $\mathbb{P}$. This is due to the fact that $\mathcal{F}_1$ just changes the weights of $\mathbb{P}$, and nothing else. Now, the continuity of $\mathcal{F}_1$ is trivial since the weight of each rule in $\mathbb{P}$ is changed only by using the continuous operator $\neg$.

Concerning $\mathcal{F}_2$, the syntactic part of $\mathbb{P}$ can be considered fixed and positive. This is due to the fact that its only inputs are of the form $\mathbb{P}_I$, therefore, the number of rules is fixed, negation does not occur in $\mathbb{P}$, and the only elements which can change are the weights. As a result, $\mathcal{F}_2$ can be seen as a function from $[0,1]^k$ to the set of interpretations. Note that this restriction over $\mathcal{F}_2$ does not disallow the composition between $\mathcal{F}_1$ and $\mathcal{F}_2$. To prove that $\mathcal{F}_2$ is continuous note, firstly, that the immediate consequence operator is continuous with respect to the weights in $\mathbb{P}$, since every operator in the definition of $T_{\mathbb{P}}$ (namely sup and $*$) is continuous. Secondly, a direct consequence of the termination result introduced in [10, see Cor. 29] ensures that if $\mathbb{P}$ is a finite positive program, then $\mathrm{lfp}(T_{\mathbb{P}})$ can be obtained by iterating finitely many times the immediate consequence operator; in other words, $\mathrm{lfp}(T_{\mathbb{P}}) = T_{\mathbb{P}}^k(I_\perp)$ where k is the number of rules in $\mathbb{P}$. Therefore, as the operator $\mathcal{F}_2$ is a finite composition of continuous operators, $\mathcal{F}_2$ is also continuous.

Finally, as $\mathcal{R}(I) = \mathrm{lfp}(T_{\mathbb{P}_I})$ is a composition of two continuous operators, $\mathcal{R}(I)$ is continuous as well. Hence we can apply Brouwer's fix-point theorem to $\mathcal{R}(I)$ and ensure that it has at least a fix-point. To conclude, we only have to note that every fix-point of $\mathcal{R}(I)$ is actually a stable model of $\mathbb{P}$. $\square$

Example 4. The existence of stable model for the normal residuated logic program below

$$\langle p \leftarrow \neg q \;; 0.8\rangle$$
$$\langle q \leftarrow \neg r \;; 0.7\rangle$$
$$\langle r \leftarrow \neg p \;; 0.9\rangle$$

is not always guaranteed. For example, if we consider the residuated lattice $L = ([0,1], *, \leftarrow, \neg)$ determined by $x * y = x \cdot y$ and

$$\neg(x) = \begin{cases} 0 & \text{if } x > 0.5 \\ 1 & \text{if } x \leq 0.5 \end{cases}$$

then the program has not stable models. However, if we consider the residuated lattice $L = ([0,1], *, \leftarrow, \neg)$ determined by $x * y = x \cdot y$ and $\neg(x) = 1 - x$ the normal residuated logic program has the following stable model

$$M = \{(p, 0.4946808); (q, 0.3816489); (r, 0.4547872)\}$$

Obviously, the sufficient condition provided in Theorem 3 is not a necessary condition. Considering the residuated lattice $L = ([0,1], *, \leftarrow, \neg)$ determined by

$$x * y = \begin{cases} x & \text{if } y = 1 \\ y & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \neg(x) = \begin{cases} 0 & \text{if } x > 0.9 \\ 1 & \text{if } x \leq 0.9 \end{cases}$$

the program above has one stable model, $M = \{(p, 0.8); (q, 0.7); (r, 0.9)\}$; although the connectives $*$ and $\neg$ are not continuous.

Remark 2. *It is important to recall that most connectives in fuzzy logic are defined on the unit interval $[0,1]$. Thus the condition about continuity on a Euclidean space as sets of truth-values is not excessively restrictive. Moreover, most t-norms used currently in fuzzy logic are continuous (Gödel, Łukasiewicz, product, ...), therefore the theorem establishes that in the most used fuzzy frameworks, the existence of fuzzy stable models is always guaranteed.*

IV. Assigning Suitable Fuzzy Stable Models to Inconsistent Classical Logic Programs

As we have seen in the previous section, stable models are guaranteed to exist when the underlying residuated lattice is $([0,1], \leq, *, \leftarrow, \neg)$ with $*$ and $\neg$ continuous. As a result, by a suitable modification of the underlying lattice of truth-values, it could be possible to assign a stable model semantics to a normal program even when the program is unstable on its original residuated structure. Although it is not difficult to imagine how this is to be done, we specify below how a crisp logic program $\mathbb{P}$ is embedded into a suitable residuated logic program framework.

A crisp logic program $\mathbb{P}$ can be seen as a residuated logic program under a residuated lattice $(L, \leq, *, \leftarrow, \neg)$ substituting each rule in $\mathbb{P}$

$$p \leftarrow q_1, \dots, q_m, \neg q_{m+1}, \dots, \neg q_n$$

by

$$\langle p \leftarrow q_1 * \dots * q_m * \neg q_{m+1} * \dots * \neg q_n \;; 1\rangle$$

It is important to point out that this translation preserves the set of existing crisp stable models when we consider the embedded program into the residuated structure, in short, we never lose existing crisp stable models but can obtain new fuzzy stable models. This is formally specified as follows:

Proposition 4. *Let $\mathbb{P}$ be a classical logic program and let M be an L-interpretation such that $M(p) \in \{0,1\}$ for every propositional symbol p. Then, M is a stable model of $\mathbb{P}$ in the classical sense if and only if M is a fuzzy stable model of $\mathbb{P}$ interpreted in the residuated lattice $(L, \leq, *, \leftarrow, \neg)$.*

Proof: The equivalence stated by this Proposition follows directly from the following two items:

- The fuzzy reduct $\mathbb{P}_M$ is equivalent to the classical reduct.
- The least fixpoint of the fuzzy immediate consequence operator $T_{\mathbb{P}_M}$ coincides with that of the classical one.

To see the equivalence between reducts, consider a rule:

$$\langle p \leftarrow q_1 * \cdots * q_m * \neg q_{m+1} * \cdots * \neg q_n \ ; 1\rangle$$

in a program $\mathbb{P}$. Then the corresponding rule in $\mathbb{P}_M$ is:

$$\langle p \leftarrow q_1 * \cdots * q_m \ ; \dot{\neg} M(q_{m+1}) * \cdots * \dot{\neg} M(q_n)\rangle$$

which is equivalent to

1) Remove the rule if M assigns 1 to some negated propositional symbol.
2) Remove the negative propositional symbols in the body of the rule.

as stated the classical reduct.

In order to prove the second item above, let I be an L-interpretation such that $I(p) \in \{0, 1\}$ for every propositional symbol p. Then, by Definition 4,

$$T_{\mathbb{P}_M}(I)(p) = \sup\{I(\mathcal{B}) * 1\colon \langle p \leftarrow \mathcal{B}; \quad 1\rangle \in \mathbb{P}_M\}$$

Note finally, that for each rule $\langle p \leftarrow \mathcal{B};\ 1\rangle \in \mathbb{P}_M$, the value $I(\mathcal{B})$ is equal to 1 if and only if for every propositional symbol q occurring in $\mathcal{B}$ we have that $I(q) = 1$; otherwise $I(\mathcal{B}) = 0$. This is equivalent to say that $T_{\mathbb{P}_M}(I)(p) = 1$ if and only if there exists one rule whose head is p and every propositional symbol q in its body satisfies $I(p) = 1$. ■

In the example below, we show the result of the embedding into the unit interval of two different programs, and analyze the resulting sets of fuzzy stable models with regard to the corresponding sets of classical stable models.

Example 5. *Consider the following crisp normal programs*

$$\begin{aligned}
\mathbb{P}_1 &= \{\langle p \leftarrow \neg q\rangle; \langle q \leftarrow \neg p\rangle\}\\
\mathbb{P}_2 &= \{\langle p \leftarrow \neg p\rangle; \langle q \leftarrow \neg q\rangle\}\\
\mathbb{P}_3 &= \{\langle p \leftarrow \neg q\rangle; \langle q \leftarrow \neg p\rangle, ; \langle r \leftarrow \neg r\rangle\}
\end{aligned}$$

Firstly, we have that $\mathbb{P}_1$ has two stable models M_1 and M_2, but $\mathbb{P}_2$ and $\mathbb{P}_3$ do not have any stable model. Specifically,

$$\begin{aligned}
SM(\mathbb{P}_1) &= \big\{\{(p, \lambda); (q, 1-\lambda)\} \mid \lambda \in \{0, 1\}\big\}\\
SM(\mathbb{P}_2) &= \varnothing\\
SM(\mathbb{P}_3) &= \varnothing
\end{aligned}$$

On the other hand, when interpreted as fuzzy logic programs in the unit interval under product logic and the standard negation operator, $\neg x = 1 - x$, then we obtain the following sets of fuzzy stable models for programs $\mathbb{P}_1$ and $\mathbb{P}_2$:

$$\begin{aligned}
FSM(\mathbb{P}_1) &= \big\{\{(p, \lambda); (q, 1-\lambda)\} \mid \lambda \in [0, 1]\big\}\\
FSM(\mathbb{P}_2) &= \big\{\{(p, 1/2), (q, 1/2)\}\big\}\\
FSM(\mathbb{P}_3) &= \big\{\{(p, \lambda); (q, 1-\lambda); (r, 1/2)\} \mid \lambda \in [0, 1]\big\}
\end{aligned}$$

Note that the residuated interpretation of program $\mathbb{P}_1$ preserves the classical stable models and, moreover, provides infinitely many other fuzzy stable models; concerning $\mathbb{P}_2$, the residuated interpretation provides just one fuzzy stable models which, moreover, turns out to "coincide" with the 3-valued interpretation of the well-founded semantics; finally, for $\mathbb{P}_3$ we obtain a set of fuzzy stable models which contains the corresponding to the well-founded semantics.

It is remarkable to note that in program $\mathbb{P}_3$ we obtain models which are "partially classical" in that some, but not all, propositional variables obtain boolean values.

We propose the use of an order relation in the set of fuzzy stable models, in order to filter out those which are closer to be classical, in the sense that assign a boolean value to a maximal set of propositional variables. The motivation for this is two-fold:

1) In the case of an inconsistent program, as our starting point is a classical logic program, we would like to consider those fuzzy stable models which are as close to be classical as possible.
2) On the other hand, a beneficial effect can be obtained when considering of partially classical models, as it might be possible to isolate parts of the program which are affected by either uncertainty or inconsistence.
   Uncertainty arises when we can assign different Boolean truth values (*true* or *false*) to a given propositional symbol, whereas inconsistence arises when we cannot assign a Boolean truth value to one propositional symbol.

We will define a pre-order relation on the set of L-interpretations, with the underlying goal of representing the notion of "being more classical":

Definition 7. *Let I and J be L-interpretations. We say that I is more classical than J (denoted by $J \sqsubseteq I$) if for every propositional symbol p such that $J(p) \in \{0, 1\}$ we have that $I(p) \in \{0, 1\}$.*

Moreover, we say that I is strictly more classical than J (denoted by $J \sqsubset I$) if $J \sqsubseteq I$ and there is a propositional symbol p such that $I(p) \in \{0, 1\}$ and $J(p) \notin \{0, 1\}$.

Note that $\sqsubseteq$ defines a pre-order relation in the set of L-interpretations since it is obviously reflexive and transitive; however, it fails to be antisymmetric. On the basis of this preorder, we can state that a stable model M_1 is accepted to the stable model M_2 if $M_2 \sqsubseteq M_1$; as a result, our accepted stable models are those M such that there is no fuzzy stable model N such that $M \sqsubset N$.

A good property of this choice of stable models is that it coincides with the classical stable model semantics when the program considered is consistent (in the sense that it admits stable models).

Corollary 1. *Let $\mathbb{P}$ be a consistent classical logic program. Then the set of accepted stable models over every residuated lattice structure coincides with the set of classical stable models.*

The following example shows that, in the case of inconsistent classical logic programs, the use of the preorder defined above, can somehow distinguish among those propositional sybols involved with uncertainty and those involved with inconsistence.

Example 6. *Continuing with Example 5, the accepted models for $\mathbb{P}_1$ turn out to be its classical stable models; for $\mathbb{P}_2$ the only choice is to consider the model $\{(p, 1/2), (q, 1/2)\}$ as the accepted stable model; finally, for $\mathbb{P}_3$, the accepted models are two "partially classical" models, namely,*

$$M_1 = \{(p,1),(q,0),(r,1/2)\}$$
$$M_2 = \{(p,0),(q,1),(r,1/2)\}$$

Note that the inconsistence of program $\mathbb{P}_3$ is due to the propositional symbol r and, interestingly enough, in both accepted stable models the non-Boolean truth-value $1/2$ is assigned only to r. On the other hand, the uncertainty related to the values of p and q arises from the fact that there exist two accepted stable models in which p and q somehow alternate their values (and we do not have reasons to prefer one to the other).

Note that the approach of building stable models in fuzzy frameworks can be applied, in principle, in every residuated lattice. However, we only can guarantee the existence of stable models in structures such as Kleene's 3-valued logic [3], in residuated lattices allowing a bilattice structure [11], [5], or residuated lattices satisfying the hypothesis of Theorem 3. The advantage of choosing the former approach instead of, for instance, Kleene's 3-valued logic is that we have infinitely many non-Boolean values to represent inconsistence.

Furthermore, note that even if we restrict our attention only to residuated lattice defined on $[0,1]$, we have a lot of possibilities to consider. Let us see a few examples to show the different information provided by three different residuated structures: minimum, product and Łukasiewicz with the standard negation.

Example 7. *Consider the following classical program*

$$\{\langle p \leftarrow \neg p\rangle; \langle q \leftarrow \neg q, p\rangle\}$$

Under minimum logic, there is just one model $M_M = \{(p, 1/2), (q, 1/2)\}$, which coincides with its well-founded model. If we consider the product logic, the program once again has just one stable model (which is accepted) $M_P = \{(p, 1/2); (q, 1/3)\}$. Finally, if we consider the Łukasiewicz connectives, the only stable model is $M_Ł = \{(p, 1/2), (q, 1/4)\}$.

Note that in the two latter cases, the truth value of p is greater than that of q, somehow denoting the fact that the rule defining q is self-contradictory and, moreover, depends on another inconsistent propositional symbol, that is, p.

Example 8. *Consider the program*

$$\{\langle p \leftarrow \neg p\rangle; \langle r \leftarrow \neg s, p\rangle; \langle s \leftarrow \neg r, p\rangle\}$$

As in the previous example, under minimum logic there is just one model namely $M_M = \{(p, 1/2), (r, 1/2), (s, 1/2)\}$, the well-founded model. If we consider the product logic, the program has once again one accepted stable model $M_P = \{(p, 1/2); (r, 1/3); (s, 1/3)\}$.

If we consider the Łukasiewicz connectives we obtain a parameterized set of stable models

$$M_\lambda = \{(p, 1/2); (r, \lambda); (s, 1/2 - \lambda)\} \colon \lambda \in [0, 1/2]$$

In this set of models, only two are accepted, namely

$$M_0 = \{(p, 1/2); (r, 0); (s, 1/2)\}$$
$$M_{1/2} = \{(p, 1/2); (r, 1/2); (s, 0)\}$$

Note that for this program, the minimum logic assigns the same non-Boolean truth-value for all propositional symbol (indicating the inconsistence of all of them); product logic assigns a different value to p than to r and s (indicating the origin of the inconsistence) and in Łukasiewicz logic, the inconsistence and uncertainty living (at same time) in r and s is isolated by providing two different accepted stable models.

Example 9. *Consider the following program*

$$\{\langle p \leftarrow \neg p\rangle; \langle q \leftarrow \neg q\rangle; \langle r \leftarrow p, q\rangle\}$$

Once again the result depends on the choice of the underlying residuated logic. The only accepted model coincides with the well-founded model, namely $M_M = \{(p, 1/2), (q, 1/2), (r, 1/2)\}$. If we consider the product logic, the only stable model (and therefore the accepted one) is $M_P = \{(p, 1/2); (q, 1/2); (r, 1/4)\}$. If we consider the Łukasiewicz connectives we obtain only the following stable model, $M_Ł = \{(p, 1/2); (q, 1/2); (r, 0)\}$.

As in previous examples, these three structures provide different accepted stable models. In Gödel structure, every propositional symbol gets the same truth value $1/2$, which indicates only the inconsistence of these propositional symbols. In product logic, the value $1/4$ assigned to r indicates that its dependance on two inconsistent propositional symbols (p and q). Finally, in Łukasiewicz logic, this difference is more drastic, assigning 0 to r as a result of its dependance on the inconsistent symbols p and q.

V. Related work

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* [12] [13] [14], *Annotated Logic* [15], *Antitonic Logic* [16], *Fuzzy Description Logic* [17] [18] and *Fuzzy Logic* [19].

The problem of non-existence of stable models has already been considered in the literature. The existing works range from the characterization of those logic programs admitting stable models [1] to the definition of alternative semantics [20], [21]; for instance, the latter defines *partial stable models*, which are actually 3-valued interpretations in order to assign a new semantics to normal programs. The present work follows this approach.

Moreover, the use of preference orderings between the newly generated stable models has already proved to be a useful approach: Fitting [11] showed that the well-founded model is the least four-valued stable model with respect a specific ordering (the knowledge ordering). So, the well-founded model can be seen as an example in which a preference is defined on the set of 3-valued stable models.

We can say that the well-founded semantics is related to this paper, in the sense that it assigns a specific stable model in a more general structure of the space of truth-values. However, as stated in the introduction, it is not possible to know, in principle, whether the third truth-value, $unknown$, represents inconsistence (non-existence of answer sets) or uncertainty with regard to the value of certain propositional symbols.

Technically, the present work considers to embed the classical program into a fuzzy framework where the existence of stable model is guaranteed. This way, we can consider not only a 3-valued [3] or a 4-valued logic [11], but any logic in which we can guarantee the existence of stable models (in our case on the unit interval by considering a fuzzy logic with continuous connectives). As a result, we do not define just one semantics but infinitely many semantics for a given classical program.

Other approaches based on alternative semantics for normal logic programs, such as the *prioritized programs* or *preferred answer sets*, in which an ordering between the rules in the program is established, are somehow related to the results in this paper. In these frameworks, an ordering is defined on the set of classical stable models according to the degree that they satisfy the preference defined among the rules of the program [22]. The similarity with our approach lies on the fact that they include an ordering between (classical) models, the difference is that in our approach we generalize the semantics by considering partial fuzzy stable models.

Another source of related work is given by the preferred semantics [23] defined for argumentative frameworks in terms of preferred extensions. This semantics coincides with the partial stable model semantics of [21] and, thus, it is a further example of embedding into a more general structure.

VI. Conclusion

Given a classical normal logic program without stable models, we propose to consider an embedding into a residuated lattice $([0,1],\leq,*,\leftarrow,\neg)$ with $*$ and $\neg$ continuous. In such a context, the resulting program admits stable models, finally the set of resulting fuzzy stable model semantics is filtered in order to maintain just the "most classical" ones.

In some sense, our proposed approach allows to identify parts of the program in terms of inconsistence and uncertainty, and follows the line of previous works in which the notion of inconsistence in fuzzy logic programs has been decomposed into two different components: incoherence and instability, see [24], [25], [26].

Acknowledgements

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

References

[1] S. Costantini, "On the existence of stable models of non-stratified logic programs," *Journal of Theory and Practice of Logic Programming*, vol. 6, no. 1-2, pp. 169–212, 2006.

[2] 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.

[3] T. Przymusinski, "Well-founded semantics coincides with three-valued stable semantics," *Fundamenta Informaticae*, vol. 13, pp. 445–463, 1990.

[4] M. Fitting, "The family of stable models," *The Journal of Logic Programming*, vol. 17, no. 2-4, pp. 197 – 225, 1993.

[5] Y. Loyer and U. Straccia, "Epistemic foundation of stable model semantics," *Journal of Theory and Practice of Logic Programming*, vol. 6, pp. 355–393, 2006.

[6] M. L. Ginsberg, "Multivalued logics: a uniform approach to reasoning in artificial intelligence," *Computational Intelligence*, vol. 4, pp. 265–316, 1988.

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

[8] 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*, 2008, pp. 260–264.

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

[10] C. Damásio, J. Medina, and M. Ojeda-Aciego, "Termination of logic programs with imperfect information: applications and query procedure," *Journal of Applied Logic*, vol. 5, no. 3, pp. 435–458, 2007.

[11] M. Fitting, "Fixpoint semantics for logic programming - a survey," *Theoretical Computer Science*, vol. 278, pp. 25–51, 1999.

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

[13] G. Wagner, "Negation in fuzzy and possibilistic logic programs," in *Logic programming and soft computing*, T. P. Martin and F. A. Fontana, Eds. Taylor & Francis, 1998, ch. 6, pp. 113–128.

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

[15] U. Straccia, "Annotated answer set programming," Istituto di Scienza e Tecnologie dell'Informazione-Consiglio Nazionale delle Ricerche, Tech. Rep. 2005-TR-51, 2005.

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

[17] T. Lukasiewicz, "Fuzzy description logic programs under the answer set semantics for the semantic web," *Fundamenta Informaticae*, vol. 82, no. 3, pp. 289–310, 2008.

[18] 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*, pp. 289–298, 2007.

[19] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir, "An introduction to fuzzy answer set programming," *Ann. Math. Artif. Intell.*, vol. 50, no. 3-4, pp. 363–388, 2007.

[20] M. Osorio, J. A. Navarro Pérez, J. R. Arrazola Ramírez, and V. Borja Macías, "Logics with common weak completions," *Journal of Logic and Computation*, vol. 16, no. 6, pp. 867–890, 2006.

[21] D. Saccà and C. Zaniolo, "Stable models and non-determinism in logic programs with negation," in *Proceedings of the ACM SIGMOD-SIGACT Symposium on Principles of Database Systems*, 1990, pp. 205–217.

[22] G. Brewka and T. Eiter, "Preferred answer sets for extended logic programs," *Artificial Intelligence*, vol. 109, pp. 297–356, 1998.

[23] P. M. Dung, "Negation as hypothesis: An abductive foundation for logic programming," in *Proceeding of the 8th International Conference on Logic Programming*, 1991, pp. 3–17.

[24] 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.

[25] ——, "Measuring instability in normal residuated logic programs: discarding information," *Communications in Computer and Information Science*, vol. 80, pp. 128–137, 2010.

[26] ——, "Measuring instability in normal residuated logic programs: adding information," in *IEEE Intl Conf on Fuzzy Systems (FUZZ-IEEE'10)*, 2010, pp. 2244–2250.