Proceedings of the 10th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2010
27–30 June 2010.

On the existence of stable models in normal residuated logic programs

Nicolás Madrid[1] **and Manuel Ojeda-Aciego**[1]

[1] *Departamento de Matemática Aplicada, Universidad de Málaga, Spain*

emails: nmadrid@ctima.uma.es, aciego@ctima.uma.es

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.

1 Introduction

Similarly to classical logic programming, the existence of fuzzy stable models cannot be guaranteed for an arbitrary normal residuated logic program [11]. Necessary conditions to ensure the existence of stable models has been widely studied in classical logic programming. In fact, the syntactic characterization of normal programs with stable models can be found in [1].

However the characterization in the fuzzy framework is much more complicated since it involves two different dimensions: "the syntactic structure of the normal program" and "the choice of suitable connectives in the residuated lattice". For short, we will call them the *syntactic* and the *semantic dimension*, respectively.

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 [12] proves that every normal logic program has stable models in the 3-valued Kleene logic and, more generally, [3, 8] show that every normal

residuated logic program has stable models if the underlying residuated lattice has an appropriate bilattice structure [5].

In this paper we provide another condition on the residuated lattice to enssure 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.

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

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* $p, p_1, \ldots, p_n$ *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, 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* $\langle \ell \leftarrow \mathcal{B}; \quad \vartheta \rangle$ *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 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* $p \in \Pi_{\mathbb{P}}$*.*

2.1 Stable Models

Our aim in this section is to recall the adaptation given in [10] of the original approach by Gelfond and Lifschitz [4] to the framework of 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:

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 that *a fuzzy interpretat*ion 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 $I(q) = 0$. □

[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.

3 The Main Result

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

Theorem 2 *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 $\mathcal{R}(I) = \text{lfp}(T_{\mathbb{P}_I})$ is continuous. Note that this operator can be seen as a composition of two operators $\mathcal{F}_1(I) = \mathbb{P}_I$ and $\mathcal{F}_2(\mathbb{P}) = \text{lfp}(T_{\mathbb{P}})$. 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 [2, see Cor. 29] ensures that if $\mathbb{P}$ is a finite positive program, then $\text{lfp}(T_{\mathbb{P}})$ can be obtained by iterating finitely many times the immediate consequence operator; in other words, $\text{lfp}(T_{\mathbb{P}}) = T_{\mathbb{P}}^k(I_\perp)$ 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 $\mathcal{R}(I) = \text{lfp}(T_{\mathbb{P}_I})$ 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 $\mathbb{P}$. □

Example 2 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 2 is not a necessary condition. Considering the residuated lattice $L = ([0,1], *, \leftarrow, \neg)$ determined by

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

the program above has one stable model, $M = \{(p, 0); (q, 0); (r, 0)\}$; although the connectives $*$ and $\neg$ are not continuous.

Remark 1 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 much 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.

4 Related Work

As stated in the introduction, one can find several conditions in the literature which guarantee the existence of stable models. Whereas in logic programming the syntactic characterization of consistent normal program was done in [1], the existence of stable models in fuzzy logic programming is an open problem; and apparently more complicated.

To the best of our knowledge, there are only other two sufficient conditions in fuzzy logic programming to guarantee the existence of stable model. The first one is given in the fuzzy description logic paradigm, and can be found in [9]. It is done at the syntactic dimension and extends a result already known in logic programming [6].

Definition 6 *A normal residuated logic program $\mathbb{P}$ is called* locally stratified *if there is a level function $||\cdot||$ such that for every rule $\langle p \leftarrow p_1 * \cdots * p_k * \neg p_{k+1} * \cdots * \neg p_n; \vartheta \rangle$ of $\mathbb{P}$:*

- $||p|| \geq ||p_i||$ *for all* $i \in \{1, \ldots, k\}$

- $||p|| > ||p_i||$ *for all* $i \in \{k+1, \ldots, n\}$

Proposition 1 *A stratified normal residuated logic program has one, and only one, stable model.*

The other result appears in [8], and is due to the use of *bilattices* as the set of truth-values. Briefly, a bilattice is a tuple $(L, \leq_t, \leq_k)$ where $(L, \leq_t)$ and $(L, \leq_k)$ form two complete lattices. Such a structure is used in [7] in order to define the well-founded semantics in fuzzy logic programming through the least stable model under the ordering $\leq_k$; i.e by generalizing a result provided in [3] for the classical case which relates the well-founded semantics and stable model semantics.

Proposition 2 *Let* $\mathbb{P}$ *be a normal logic program defined over the residuated lattice* $(L, \leq, *, \leftarrow, \neg)$*. If there is an ordering* $\leq_k$ *such that:*

- $(L, \leq, \leq_k)$ *is a bilattice*
- $*$ *and* $\neg$ *are monotonic w.r.t.* $\leq_k$

then, there exists at least one stable model of $\mathbb{P}$*.*

The key-point of proposition 2 is to find an ordering over L such that $*$ and $\neg$ are monotonic with respect it, thus we can to asure the existence of stable model.

5 Future Work

The result of Theorem 2 has interesting potential applications. To begin with, we can avoid the inconsistency in fuzzy logic programs by using continuous connectives. Moreover, the result is useful to resolve inconsistencies of normal programs defined on a linear lattice by extending them over $[0, 1]$ with continuous connectives. For example, consider the following normal program in classical logic programming:

$$r_1 : p \leftarrow \neg q, s \qquad r_2 : r \leftarrow \neg t, \neg p$$
$$r_3 : q \leftarrow \neg r \qquad r_5 : s \leftarrow$$
$$r_5 : u \leftarrow \neg t, s \qquad r_6 : v \leftarrow \neg v, \neg r$$

where "," denotes the classical conjunction. Clearly the program is inconsistent but we can assign a fuzzy stable model semantics by embedding the program into a residuated lattice $([0, 1], *, \leftarrow, \neg)$. For example, consider the connectives $x * y = x \cdot y$ and $\neg(x) = 1 - x$, then the program above (substituting "," by "$*$" and including the weight 1 in each rule) has the following stable model:

$$M = \{(p, 0.5); (q, 0.5); (r, 0.5); (s, 1); (t, 0); (u, 1); (v, 1/3)\}$$

Notice that if we collapse each $x \in (0, 1)$ to one undefined value (that is, M assigns to p, q, r and v the same truth-value "undefined") the semantics is equivalent to the well-founded semantics. Notice, however, that the residuated semantics is slightly more expressive, due to its ability to assign any value in the unit interval.

Acknowledgements

This work has been partially supported by Junta de Andalucía grant P09-FQM-5233, and by the EU (FEDER), and the Spanish Science Ministry under grant TIN2009-14562-C05-01.

References

[1] S. Costantini. On the existence of stable models of non-stratified logic programs. *Journal of Theory and Practice of Logic Programming*, 6(1-2):169–212, 2006.

[2] C. Damásio, J. Medina, and M. Ojeda-Aciego. Termination of logic programs with imperfect information: applications and query procedure. *Journal of Applied Logic*, 5(3):435–458, 2007.

[3] M. Fitting. The family of stable models. *The Journal of Logic Programming*, 17(2-4):197 – 225, 1993.

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

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

[6] J. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.

[7] Y. Loyer and U. Straccia. The well-founded semantics in normal logic programs with uncertainty. *Lect. Notes in Computer Science*, 2441:152–166, 2002.

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

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

[10] 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.

[11] 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.

[12] T. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundamenta Informaticae*, 13:445–463, 1990.