TOWARDS A FUZZY ANSWER SET SEMANTICS FOR RESIDUATED LOGIC PROGRAMS

Nicolás Madrid Labrador, Manuel Ojeda-Aciego
Dept. Matemática Aplicada. Universidad de Málaga
{nmadrid,aciego}@ctima.uma.es

Abstract

In this work we introduce the first steps towards the definition of an answer set semantics for residuated logic programs with negation.

Keywords: Fuzzy logic programming, Stable models, Answer Sets.

1 Introduction

Twenty years later of the introduction of the stable model semantics in classical logic programming, we are still seeing a number of different applications of this notion in different logical frameworks.

Answer set semantics is an intuitive and elegant generalisation of the stable model semantics which provides a powerful solution for knowledge representation and non-monotonic reasoning problems. Originally, they were intended to deal with two negations, one strong negation and default negation. The use of these two types of negation is advocated in many contexts of interest, in particular in [12] there use is justified in relation to web rules.

It is convenient to note that stable models were initially aimed at formalizing the use of negation in logic programming as negation-as-failure and, thus, are closely related to reasoning under uncertainty. For instance, the closed world assumption for a given predicate P allows for extracting negative knowledge about P from the absence of positive information about it.

The ideal environment for developing a theory of management of uncertainty is fuzzy logic. Therefore, fuzzy logic programming has become a target theory for a suitable generalization of answer set semantics.

In this paper, we focus on the initial definitions of stable model and answer set in the framework of residuated logic programs [2], as a initial step towards an answer set semantics for multi-adjoint logic programs, which were introduced in [8].

2 Preliminaries

In this section we include the definitions needed to introduce our approach to residuated logic programs with negation. Let us start with the definition of residuated lattice:

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

1. *$(L, \leq)$ is a complete and bounded lattice with largest element 1 and least element 0.*
2. *$(L, *, 1)$ is a commutative monoid unit element 1.*
3. *$*$ and $\leftarrow$ form an adjoint pair, i.e:*

$$z \leq (x \leftarrow y) \text{ iff } y * z \leq z \quad \text{for all } x, y, z \in L.$$

If the infimum in L can be defined in terms of $*$ and $\leftarrow$ as $x \wedge y = x * (y \leftarrow x)$, then the operators $*$ and $\leftarrow$ behave like a generalized conjunction (increasing in every argument) and a generalized implication (decreasing in the antecedent and increasing in the consequent), see [7].

In the rest of the paper we will consider two different types of operators of negation, which will modelize strong negation $\sim$ and default negation $\neg$. The former negation operator will be used in order to define a kind of consistency for a fuzzy interpretation, whereas the latter will be used during the construction of a reduct (or program division).

Definition 2 *A* residuated lattice with negation *is an algebra $(L, *, \leftarrow, \sim, \neg)$, where $(L, *, \leftarrow)$ is a residuated lattice, $\sim$ is a negation operator and $\neg$ is a default negation operator.*

In order to introduce our logic programs, we will assume a set Π of propositional symbols. If $p \in \Pi$, then both p and $\sim p$ are called *literals*. We will denote arbitrary literals with the symbol ℓ (possible subscripted), and the set of all literals as Lit.

Definition 3 *Given a residuated lattice with negation $(L, *, \leftarrow, \sim, \neg)$, an* extended residuated logic program *$\mathbb{P}$ is a set of weighted rules of the form*

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

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

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

Definition 4 *A* (fuzzy) interpretation *is a mapping $I \colon Lit \to L$; note that the domain of an interpretation can be extended to any rule by homomorphic extension.*

We say that I satisfies *a rule $\langle \ell \leftarrow \mathcal{B}; \vartheta \rangle$ 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 domain of our interpretations is the whole set of literals Lit, hence we are following an approach which is not compositional with regard to either the strong negation or, as usual, default negation.

An extended general residuated logic program $\mathbb{P}$ is said to be:

- *definite* if it does not contain negation operators.
- *general* if it does not contain strong negation, but might contain default negation.
- *normal* if it does not contain default negation but it might contain strong negation.

3 Fuzzy stable sets

Our aim in this section is to adapt the approach given in [5] to the general residuated logic programs just defined in the section above.

Let us consider a general residuated logic program $\mathbb{P}$ together with a fuzzy interpretation I. To begin with, we will construct a new general 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]

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

Notice that the new program $\mathbb{P}_I$ is general, that is, does not contain default 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 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 $\mathbb{P}_J$.

In the following lemma, we show that every model M of the program $\mathbb{P}$ is a model of the reduct $\mathbb{P}_M$.

Lemma 1 *M is a model of $\mathbb{P}$ iff M is a model of $\mathbb{P}_M$.*

Proof Let us consider a rule in $\mathbb{P}$

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

Assume that M is a model of $\mathbb{P}$, then M satisfies the rule above, that is

$$M(p_1 * \cdots * p_m * \neg p_{m+1} * \cdots * \neg p_n) * \vartheta \leq I(p)$$

Now, by the homomorphic extension of interpretations and the adjoint property, the inequality above is equivalent to:

$$\neg M(p_{m+1}) * \cdots * \neg M(p_n) * \vartheta \leq M(p \leftarrow p_1 * \cdots * p_m)$$

that is, M satisfies the rule

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

□

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 6 *Let $\mathbb{P}$ be an extended residuated logic program and let I be a fuzzy 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 $\mathbb{P}$.*

[1]Note the use of a dotted version of the negation symbol to denote the semantical negation operator.

Proof Let M be a stable set of $\mathbb{P}$ which, by definition, is a model of $\mathbb{P}_M$. Now, by Lemma 1, M is a model of $\mathbb{P}$. In order to prove minimality, assume that N is another model of $\mathbb{P}$ such that $N \leq M$, and let us show that $N = M$. Let us note the following facts about N:

1. N is a model of $\mathbb{P}_N$, again by Lemma 1.
2. The weights in the rules in $\mathbb{P}_N$ are greater or equal than those in $\mathbb{P}_M$, by the remark to Definition 5.

As a result, since N is a model of $\mathbb{P}_N$, then N is a model of $\mathbb{P}_M$ as well.

Finally, as M is a minimal model of $\mathbb{P}_M$, then $M \subseteq N$. $\square$

Thanks to Theorem 1 we know that every stable set is a model, thus we have a suitable generalization of the concept of stable model in this generalized framework. Hereafter, specially in a semantic context, we will use the term stable model to refer to a stable set.

This approach to reducts and stable models is a conservative extension of the classical approach, as we show below:

Firstly, a classical (bi-valued) extended program $\mathbb{P}$ can be seen as an extended residuated logic program on the lattice $\{0, 1\}$ and such that each rule in $\mathbb{P}$ has weight 1.

Now, let us see that the construction of our (fuzzy) reduct on a classical program provides exactly the classical reduct.

Consider a rule in $\mathbb{P}$, which can be written as follows:

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

Now, given a fuzzy interpretation I, the rule is transformed in the reduct in the rule $p \leftarrow p_1 \wedge \cdots \wedge p_n$ together with a new weight. Obviously, the value of this weight is either 0 or 1, as follows:

- if $p_i \notin M$ for all $i \in \{m+1, ..., n\}$, the assigned value of the rule is

  $$1 * \neg p_{m+1} * \cdots * \neg p_n = 1 * \neg 0 * \cdots * \neg 0 = 1$$

- if $p_i \in M$ for some $i \in \{m+1, ..., n\}$; then the formula for constructing the weight in the reduct leads to the value 0.

As a result, the rule is transformed in either

$$\langle p \leftarrow p_1 \wedge \cdots \wedge p_m; \quad 1\rangle$$

or

$$\langle p \leftarrow p_1 \wedge \cdots \wedge p_m; \quad 0\rangle$$

Note that the effect of the weight 0 is similar to eliminate the rule, the same as in the classical approach.

In the following example we use a simple general residuated program with just one rule in order to show some subtle differences generated by the extension to the fuzzy case:

Example Let us consider the following program with just one rule $\mathbb{P} = \{p \leftarrow \neg q\}$. In classical logic, for this program there exist exactly four different interpretations, and only one of them is a stable model, namely, $I(p) = 1$ and $I(q) = 0$.

Now, let us consider the extended version of the program above, where the only difference is that we can assign a weight to the rule and that the propositional symbols are evaluated in a residuated lattice with negation

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

Given a fuzzy interpretation $I \colon \Pi \to L$, the reduct $\mathbb{P}_I$ is the rule (actually, the fact) below

$$\langle p; \quad \vartheta * \dot{\neg} I(q)\rangle$$

for which the least model is $M(p) = \vartheta * \dot{\neg} I(q)$, and $M(q) = 0$. As a result, I is a stable model of $\mathbb{P}$ if and only if $I(p) = \vartheta * \dot{\neg} I(0)$. $\square$

4 Fuzzy answer sets

In this section, we concentrate on strong negation and we will consider normal residuated logic programs.

Note that, as our interpretations are defined on the set of literals, every normal program has a least model which can be obtained, for instance, by iterating the immediate consequence operator, see [2].

In the classical case, one has to take into account the interaction between opposite literals in order to reject inconsistent models. The advantage of working in a fuzzy framework is that one can allow that two opposite literals, such as p and $\sim p$, live together ... under some requirements.

Our approach will be based on a generalization of the concept of consistency which we have called *coherence*, to distinguish it from other existing definitions of consistency in a fuzzy setting.

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

It is easy to check that our notion of coherence coincides with consistency when applied in the framework of classical logic.

The following properties of coherent interpretations will be used in the rest of the section.

Proposition 1 *Let I and J be interpretations such that $I \leq J$ and J is coherent, then I is coherent as well.*

Proof Let p be a propositional symbol, then by the inequality $I \leq J$, the coherence of J and the decreasing property of $\dot{\sim}$ we have

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

□

Corollary 1 *If M is a fuzzy coherent model of $\mathbb{P}$ and $T \leq M$ is a another model of $\mathbb{P}$ then T is a coherent model as well.*

As a consequence of the previous corollary we can introduce the following definition:

Definition 8 *Let $\mathbb{P}$ be a normal residuated logic program, we say that $\mathbb{P}$ is* coherent *if its least model is coherent.*

Example Consider the following normal residuated logic program over the unit interval and strong negation $\sim x = 1 - x$:

$$\begin{aligned} \langle p \leftarrow; &\quad 1\rangle \\ \langle \sim p \leftarrow; &\quad 0'3\rangle \end{aligned}$$

This program is not coherent because its unique minimal model $M = \{(p, 1), (\sim p, 0'3)\}$ is not a coherent interpretation, since

$$0'3 = M(\sim p) > \sim M(p) = 0$$

□

Once the concept of coherence has been presented, we can introduce the notion of *fuzzy answer set*. Such a set is a fuzzy set of literals, similarly to the classical case, which sometimes will be considered a fuzzy interpretation.

Definition 9 *Let $\mathbb{P}$ be a coherent normal residuated logic program; a* fuzzy answer set *of $\mathbb{P}$ is its least coherent model of $\mathbb{P}$.*

If $\mathbb{P}$ is a positive program, then it is a coherent program and the fuzzy answer set of $\mathbb{P}$ is simply the least fuzzy model of $\mathbb{P}$.

Now, the definition of *fuzzy answer set* for extended residuated logic programs is just a combination of that for normal programs and stable models, via the construction of reducts [6].

Given an *extended* residuated logic program $\mathbb{P}$, it will be transformed into a new *general* logic program $\mathbb{P}^+$ in which we simply "forget" that the rules in the program are constructed from literals, and consider negative literals as new, independent, propositional symbol.

Formally, for any propositional symbol p occurring in $\mathbb{P}$, let p' be a new propositional symbol, which will be called the *positive form* of the negative literal $\sim p$. Every positive literal is, by definition, its own positive form. The positive form of the literal ℓ will be denoted by ℓ^+ and $\mathbb{P}^+$ will stand for the normal program obtained from $\mathbb{P}$ by replacing each rule

$$\ell \leftarrow \ell_1 * \ell_2 * ... * \ell_m * \neg\ell_{m+1} * \cdots * \neg\ell_n$$
by
$$\ell^+ \leftarrow \ell_1^+ * \ell_2^+ * ... * \ell_m^+ * \neg\ell_{m+1}^+ * \cdots * \neg\ell_n^+$$

The transformation above can be easily applied to fuzzy interpretations, in that, an interpretation I such that $I(\sim p) = \vartheta$ is transformed into a fuzzy interpretation I^+ such that $I^+(p') = \vartheta$. As a consequence, we obtain in a straightforward way the following lemma.

Lemma 2 *M is a model of $\mathbb{P}$ if and only if M^+ is a model of $\mathbb{P}^+$.*

It is easy to see that the program $\mathbb{P}^+$ is a general program and, thus, we can consider its stable models. This leads to our definition of fuzzy answer set for an extended program.

Definition 10 *Given an extended residuated logic program $\mathbb{P}$, a* fuzzy answer set *for $\mathbb{P}$ is a stable model for $\mathbb{P}^+$.*

Now, it is convenient to show that the different definitions made for particular classes of programs coincide (whenever it makes sense to establish such a comparison).

To begin with, in the following proposition, we state the relationship between the concepts of fuzzy answer sets for a normal program and stable model for a definite program.

Proposition 2 *Let $\mathbb{P}$ be a normal residuated logic program and I a fuzzy coherent interpretation; then I is a fuzzy answer set of $\mathbb{P}$ if and only if I^+ is a stable model of $\mathbb{P}^+$.*

Proof Follows from the fact that, if $\mathbb{P}$ is normal, then $\mathbb{P}^+$ does not contain default negation; as a result, its only stable model is its least model. □

It is clear, on the other hand, that the concept of fuzzy stable model for a general program coincides with the least model of a definite one; simply, because the latter does not contain negation (neither strong nor default).

At this point it should be clear that there exists another "reasonable" possibility for defining a fuzzy answer set for an extended program; namely, the construction of reduct[2] to be directly applied on an extended program, $\mathbb{P}^I$, and then, compute its least model. If this least model coincides with I. The natural question here is whether I is a fuzzy answer set for $\mathbb{P}$, in the sense of Definition 10.

Lemma 3 *Let $\mathbb{P}$ be an extended residuated logic program, then $(\mathbb{P}^I)^+ = (\mathbb{P}^+)_{I^+}$.*

Proof Let us consider a rule of $\mathbb{P}$

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

when constructing its reduct wrt I we obtain

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

whose positive form is

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

this is the resulting rule under $(\mathbb{P}^I)^+$.

On the other hand, the positivisation of the initial rule is

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

and when computing its reduct wrt I^+ we obtain

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

which is the resulting rule under $(\mathbb{P}^+)_{I^+}$.

Finally, note that, by definition, $\dot\neg(I^+(\ell_i)) = \dot\neg(I(\ell_i^+))$ for all $i = \{1, ..., n\}$. Therefore the rules are the same under both combinations. $\square$

From the above lemma, we can compute a fuzzy answer set for an extended logic program $\mathbb{P}$ by means of, abusing a little bit of terminology, the *extended stable models.*

Formally, we have the following theorem:

Theorem 2 *Let $\mathbb{P}$ a extended residuated logic program. A coherent fuzzy interpretation I is a fuzzy answer set of $\mathbb{P}$ if and only if I is an extended stable of $\mathbb{P}$.*

Proof : Straightforward from Lemma 3. $\square$

[2]Note that we write $\mathbb{P}^I$ to note that $\mathbb{P}$ needs not be a general program; anyway, the construction of the reduct is the same.

5 Other approaches to fuzzy answer set programming

Several approaches have been developed in order to cope with negation in a fuzzy logic programming setting. To the best of our knowledge, one of the first studies of negation in fuzzy logic programming was introduced in [11], which developed a compositional approach to encompass strong negation, in opposition to the well-known but non-compositional approach of [3].

In [4] a foundation for fuzzy logic programming was developed, but his approach was not based on implication rules, but on Horn clauses on which negation is interpreted as default negation. Simultaneously, [1] introduced the so-called antitonic logic programs, for which a well-founded and a stable model semantics was developed; antitonic logic programs are based, as our programs, on a residuated lattice, but their rules must satisfy the condition that their bodies should be either increasing in all variables or decreasing in all variables and, thus, they cannot accomodate rules containing both positive and negative literals in their bodies. Moreover, they consider partial interpretations, in the form of pairs of interpretations in order to keep track of the information about truth and non-falsity.

There are other approaches based on annotated multiple-valued logic [9]. In this paper, the underlying truht-values set is a complete lattice, and the main difference is that interpretations assign an *interval* to literals (in a manner similar to [1]).

More recently, a general introduction to fuzzy answer set programming has been introduced in [10], however their approach is not directly applicable to residuated logic programs, since the programs they considered did not have weights.

6 Conclusions and future work

In this paper we have set the initial definitions in order to start a thorough study of the answer set semantics of extended residuated logic programs. The definitions of fuzzy stable model for general program and of fuzzy answer set for a coherent normal program are used in order to provide the more general definition of fuzzy answer set for an extended fuzzy program. Finally, we conclude with the result that fuzzy answer sets can, equivalently, be computed by means of extended stable models.

A number of issues still have to be studied: for instance, the epistemological implications of the concept of coherence. In this paper, we have only taken into account that the resulting fuzzy answer sets should be validated against some consistency-related notion. Future work, should go towards imbricating this notion with threshold computation which turns out to be an important issue for negation-as-failure. For instance, the absence of evidence of p could be interpreted that the value of p is at most a threshold value which cannot be detected by the sensors which provide our information.

Finally, it is important to relate our approach with other existing approaches, and study their possible interactions, advantages and disadvantages.

References

[1] C. V. Damásio and L. M. Pereira. Antitonic logic programs. In *Proc. Logic Programming and Nonmonotonic Reasoning, LPNMR'01*, pages 379–392. Lect. Notes in Artificial Intelligence, 2173, Springer-Verlag, 2001.

[2] C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pages 748–759. Lect. Notes in Artificial Intelligence, 2143, 2001.

[3] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. Gabbay et al, editor, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 439–513. Clarendon Press, 1994.

[4] R. Ebrahim. Fuzzy logic programming. *Fuzzy Sets and Systems*, 117:215– 230, 2001.

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

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

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

[8] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146(1):43–62, 2004.

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

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

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

[12] G. Wagner. Web rules need two kinds of negation. In *Principles and Practice of Semantic Web Reasoning*, volume 2901 of *Lecture Notes in Computer Science*, pages 33–50, 2003.