2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Towards a fuzzy answer set semantics for residuated logic programs

Nicolás Madrid
Dept. Matemática Aplicada.
Universidad de Málaga
nmadrid@ctima.uma.es

Manuel Ojeda-Aciego
Dept. Matemática Aplicada.
Universidad de Málaga
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.

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 nonmonotonic reasoning problems. Originally, answer sets 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 [15] their use is justified in relation to web rules.

The overall framework of answer set programming has important links with description logics and, hence, with the semantic web, as stated in [5, 7, 9, 10]. This new paradigm requires the introduction of new layers of semantics (ontologies, rules, logic, proofs) enriching the data stored in the classical web; as a result, the problems underlying the semantic web are, in some sense, similar to those of logic programming: the logical organization of knowledge.

The present work can be seen as related to the framework of query answering in the presence of uncertainty. 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 [11].

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 tuple* $((L, \leq), *, \leftarrow)$ *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:*

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

In residuated lattices one can interpret the operator $*$ like a conjunction and the operator $\leftarrow$ like an implication.

In the rest of the paper we will consider a residuated lattice enriched with two negation operators, $(L, *, \leftarrow, \sim, \neg)$. The two negations will modelize the notions of strong negation $\sim$ and default negation $\neg$ 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 $n(1) = 0$.

The difference between strong and default negation in our context is essentially semantical, and relates to the method we use to infer the truth value of one negated propositional symbol: The strong negation operator will be used in order to define a measure of consistency of a fuzzy L-interpretation, whereas the default negation will be used during the construction of a reduct (or program division).

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.357

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

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

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 3 *A fuzzy* L-interpretation *is a mapping $I\colon Lit \to L$; note that the domain of the interpretation can be lifted to any rule by homomorphic extension.*

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 domain of our interpretations is the whole set of literals Lit, hence we are following an approach which is compositional wrt the default negation, whereas *needs not be* compositional wrt the strong negation.

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

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

3. Fuzzy stable sets

Our aim in this section is to adapt the approach given in [6] 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]

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

[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.

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 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, 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.

Remark This approach to reducts and stable models is a conservative extension of the classical approach.

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

Example Consider the following program with just one rule $\mathbb{P} = \{p \leftarrow \neg q\}$. In classical logic, for this program there are 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 general version of the program-rule above, $\langle p \leftarrow \neg q; \quad \vartheta \rangle$, 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.

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 $I(q) = 0$. □

4. Fuzzy answer sets

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

Note that, as our interpretations are defined on the set of literals, every extended 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 6 *A fuzzy L-interpretation I over Lit is* coherent *if the inequality $I(\sim p) \leq \sim I(p)$ holds for every propositional symbol p.*

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

The following properties, regarding the pointwise ordering between interpretations, will be used in the rest of the section.

Proposition 1 *Let I and J be two fuzzy L-interpretations satisfying $I \leq J$. If J is coherent, then I is coherent as well.*

Corollary 1 *If M is a fuzzy coherent model of $\mathbb{P}$, then any other model T such that $T \leq M$ is a coherent model.*

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

Definition 7 *Let $\mathbb{P}$ be an extended residuated logic program, we say that $\mathbb{P}$ is* coherent *if its least model is coherent.*

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

$$\langle p \leftarrow; \quad 1\rangle \qquad \langle \sim p \leftarrow; \quad 0'3\rangle$$

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 L-interpretation.

Definition 8 *Let $\mathbb{P}$ be a coherent extended residuated logic program; the* fuzzy answer set *of $\mathbb{P}$ is its least coherent model of $\mathbb{P}$.*

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

Now, the notion of *fuzzy answer set* for general residuated logic programs is just a combination of that for extended programs and stable models, via construction of reducts.

A *general* residuated logic program $\mathbb{P}$ 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 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 extended 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 L-interpretations, in that, an interpretation I such that $I(\sim p) = \vartheta$ is transformed into a fuzzy L-interpretation I^+ such that $I^+(p') = \vartheta$. As a consequence, we obtain in a straightforward way the following lemma.

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

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

Definition 9 *Given a general 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 an extended program and stable model for a positive program.

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

It is clear, on the other hand, that the concept of fuzzy stable model for a normal program coincides with the least model of a positive 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 a general program; namely, the construction of reduct[2] to be directly applied on a general 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 9.

Lemma 2 *Let $\mathbb{P}$ be a general residuated logic program, then $(\mathbb{P}^I)^+ = (\mathbb{P}^+)_{I^+}$.*

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

Formally, we have the following theorem:

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

5. A detailed example

Let us assume that we are the local chairs of a conference, and we want to provide information about hotels close to the venue. The degree of closeness of each hotel will be made in terms of its quality and its reachability which, in turn, is determined regarding its distance to the venue and the existence of public transport. The conference will be held in the new buildings of our department but it is not clear that the new tube station will be operative.

The rules below define a fuzzy predicate Unlikely(x) which assigns a degree of unlikeliness to each hotel in terms of how far is from the venue and the existence of tube connection.

$$\begin{array}{lll} \text{Unlikely(x)} \leftarrow \text{Tube(x), Far(x)} & ; & 0.3 \\ \text{Unlikely(x)} \leftarrow \neg\text{Tube(x), Far(x)} & ; & 0.5 \\ \text{Unlikely(x)} \leftarrow \neg\sim\text{Tube(x), Far(x)} & ; & 0.5 \\ \text{Unlikely(x)} \leftarrow \sim\text{Tube(x), Far(x)} & ; & 0.7 \end{array}$$

$$\text{GoTo(x)} \leftarrow \text{Quality(x)}, \neg\text{Unlikely(x)} \quad ; \quad 0.8$$

Far(A) ; 0.2	Quality(A) ; 0.4
Far(B) ; 0.4	Quality(B) ; 0.6
Far(C) ; 0.6	Quality(C) ; 0.8

If we are not assuming information about the existence of tube connection, the only fuzzy stable model assigns

Unlikely(A) ; 0.1	Unlikely(B) ; 0.2	Unlikely(C) ; 0.3
GoTo(A) ; 0.288	GoTo(B) ; 0.384	GoTo(C) ; 0.448

But consider that we take into account the information available about the likeliness that that tube is finished by the dates of the conference (according to the builder), as well as its unlikeliness (according to the newspapers), and incorporate the following facts:

Tube(A) ; 0.7	Tube(B) ; 0.5	Tube(C) ; 0.6
$\sim$Tube(A) ; 0.4	$\sim$Tube(B) ; 0.4	$\sim$Tube(C) ; 0.9

In this case, the new results that we obtain are the following

Unlik.(A) ; 0.06	Unlik.(B) ; 0.12	Unlik.(C) ; 0.378
GoTo(A) ; 0.300	GoTo(B) ; 0.422	GoTo(C) ; 0.398

in which we see that Hotel C is penalized due to the high risk that the tube connection is not ready.

Of course, one could argue that the assumed values for the existence or non-existence of tube are greatly contradictory and, thus, the result is questionable.

In this respect, we have to say that, certainly, the tube-related information is not coherent *regarding the standard negation*, but it is possible to assume a suitable algebraic negation operator with respect to which the resulting output is a coherent fuzzy answer set.

6. Related approaches to fuzzy ASP

Several approaches have been developed in order to cope with negation in a fuzzy logic programming setting. In this section we briefly overview some of them although, due to lack of space, a thorough comparison cannot be included and is subject of future work.

To the best of our knowledge, one of the first studies of negation in fuzzy logic programming was introduced in [14], which developed a compositional approach to encompass strong negation (one in which interpretations are homomorphic with respect to strong negation, that is, the equality $I(\sim p) = \sim I(p)$ holds for all interpretation I), 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. This use of two interpretations could be related to our use of interpretations from the set of literals $I\colon Lit \to L$, as the union of a pair of interpretations, one for the positive literals and another one for the negative literals.

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

There are other works with the similar spirit, such as the approach to answer sets based on annotated multiple-valued logic [12]. In this paper, the underlying truth-values set is a complete lattice, and the main difference is that interpretations assign an *interval* to literals (similar to [1]).

The rest of approaches included in this section have not been compared thoroughly with our approach, and their relationship with our contribution is subject of future work:

An approach to fuzzy answer set programming has been introduced in [13] which allows a very form of the logic programs and, thus, provides a parametric highly configurable framework to meet the needs of the user. In principle, this approach is not directly applicable to residuated logic programs, since the programs they considered did not have weights, but a thorough comparison with our approach is still missing.

Finally, there are several works in which answer set semantics is applied to fuzzy description logic. Regarding its application to the semantic web we have, for instance, the fuzzy dl-programs in [10] which generalize, on the one hand, the fuzzy disjunctive logic programs and, on the other hand, fuzzy description logics. This framework is further extended in [9], with the consideration of *normal* fuzzy dl-programs as a generalization of normal description logic programs under the answer set semantics by fuzzy vagueness and imprecision in both the description logic and the logic program component.

7. 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 general residuated logic programs. The definitions of fuzzy stable model for normal program and of fuzzy answer set for a coherent extended program are used in order to provide the definition of fuzzy answer set for a general fuzzy program. Finally, we conclude with the result that fuzzy answer sets can, equivalently, be computed by means of general stable models.

A number of issues still have to be studied within this research line. In this paper, we have only taken into account that the resulting fuzzy answer sets should be validated against some consistency-related notion, namely, coherence:the fact that the choice of an adequate algebraic operator for strong negation is crucial for the coherence of the resulting fuzzy answer sets is an important aspect related to future work on this topic which, due to lack of space, will be studied elsewhere. Moreover, it is interesting to know better the epistemological implications of the concept of coherence.

Other piece of future work should lead 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.

Last but not least, 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. *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 Proc. *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. 1994.

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

[5] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In *Principles of Knowledge Representation and Reasoning (KR'04)*, pages 141–151, 2004.

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

[7] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Open answer set programming for the semantic web. *Journal of Applied Logic*, 5(1):144–169, 2007.

[8] S. Heymans and D. Vermeir. Integrating semantic web reasoning and answer set programming. In *Answer Set Programming*, pages 195–209, 2003.

[9] T. Lukasiewicz. Fuzzy Description Logic Programs under the Answer Set Semantics for the Semantic Web. *Fundamenta Informaticae*, 82(3), 289-310, 2008.

[10] T. Lukasiewicz and U. Straccia. Tightly Integrated Fuzzy Description Logic Programs Under the Answer Set Semantics for the Semantic Web. In Proc. *Web Reasoning and Rule Systems*. Lecture Notes in Computer Science, 4524:289–298, 2007

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

[12] U. Straccia. Annotated answer set programming. Technical Report 2005-TR-51, Istituto di Scienza e Tecnologie dell'Informazione-CNR, 2005.

[13] 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.

[14] 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.

[15] G. Wagner. Web rules need two kinds of negation. In Proc. *PPSWR'03*, pages 33–50. Lecture Notes in Computer Science 2901, 2003.