Ann Math Artif Intell (2007) 50:363–388
DOI 10.1007/s10472-007-9080-3

An introduction to fuzzy answer set programming

Davy Van Nieuwenborgh · Martine De Cock · Dirk Vermeir

Published online: 21 July 2007

Abstract In this paper we show how the concepts of answer set programming and fuzzy logic can be successfully combined into the single framework of fuzzy answer set programming (FASP). The framework offers the best of both worlds: from the answer set semantics, it inherits the truly declarative non-monotonic reasoning capabilities while, on the other hand, the notions from fuzzy logic in the framework allow it to step away from the sharp principles used in classical logic, e.g., that something is either completely true or completely false. As fuzzy logic gives the user great flexibility regarding the choice for the interpretation of the notions of negation, conjunction, disjunction and implication, the FASP framework is highly configurable and can, e.g., be tailored to any specific area of application. Finally, the presented framework turns out to be a proper extension of classical answer set programming, as we show, in contrast to other proposals in the literature, that there are only minor restrictions one has to demand on the fuzzy operations used, in order to be able to retrieve the classical semantics using FASP.

Keywords Answer set programming · Fuzzy logic

Mathematics Subject Classifications (2000) 68T27 · 68T30 · 68T37

D. Van Nieuwenborgh (✉) · D. Vermeir
Department of Computer Science, Vrije Universiteit Brussel, VUB,
Pleinlaan 2, 1050 Brussels, Belgium
e-mail: dvnieuwe@vub.ac.be

D. Vermeir
e-mail: dvermeir@vub.ac.be

M. De Cock
Department of Applied Mathematics and Computer Science, Universiteit Gent,
UGent, Krijgslaan 281 (S9), 9000 Ghent, Belgium
e-mail: martine.decock@ugent.be

1 Introduction

The answer set programming (ASP) paradigm [15] has gained a lot of popularity in the last years, due to its truly declarative non-monotonic semantics, which has been proven useful in a number of interesting applications, e.g. [12, 16, 20, 23]. The idea behind the answer set semantics, a generalization of the stable model semantics [14], is both intuitive and elegant. Given a program P and a candidate answer set M, one computes a reduct program P^M of a simpler type for which a semantics $(P^M)^\star$ is known. The reduct P^M is obtained from P by taking into account the consequences of accepting the proposed truth values of the literals in M. The candidate set M is then an answer set just when $(P^M)^\star = M$, i.e. M is "self-producible."

An alternative characterization of answer sets is given in [22, 26] in terms of unfounded sets. Intuitively, an unfounded set is a set of literals for which there is no motivation to suppose that these literals have to be true. The reason herefore is that these literals either depend on each other, or the rules that can motivate them are not applicable. A candidate answer set M is then an answer set of P iff it is a model of P and it does not contain such unfounded sets.

Although ASP provides a powerful solution for knowledge representation and non-monotonic reasoning, it has some drawbacks regarding the configurability of the semantics w.r.t. the type of application under consideration, as witnessed by the large number of extensions, both syntactically and semantically, that have been proposed in the literature [2, 5, 7, 10]. E.g., most[1] ASP semantics demand that a solution to a program satisfies all the rules. Further, the literals available in the program, i.e. the building blocks of rules, can only be true or false (or unknown when one considers e.g. the well-founded semantics [27]), and classical consistency is mandatory, i.e. a and $\neg a$ cannot be true at the same time (or not even "a bit" true at the same time). Also the interpretation of negation as failure, the construct that gives ASP its non-monotonicity, is very sharp: not a is true iff a is not true.

Sometimes however, one wants to work with concepts that are not easily categorized as being either true or false. Consider e.g. the atoms *shiver*, *high_temperature* and *sick* in the following rule

$$sick \leftarrow high\_temperature, shiver \ .$$

In the traditional setting this implies that a person is sick if she suffers from shivering and has a high temperature. However, when is a temperature to be considered "high" and when can one say that someone is shivering? Since in ASP literals can only be true or false, these problems would be tackled by imposing thresholds, e.g. stating that *high_temperature* is true if the temperature is at least 38°, and false otherwise. The choice of the exact values of such thresholds is often quite arbitrary but nevertheless has a large impact on the outcome. A temperature variation as small as 0.1° can make the difference in whether or not the rule is applicable. Doctors and patients on the other hand do not perceive such a dramatic difference between a temperature of 37.9° and one of 38°; they rather perceive a gradual transition instead of an abrupt one. To model this, one should be able to express that literals are true to a certain extent, as opposed to being either true or false.

[1] Some semantics dealing with preferences on rules [6, 13, 28] are more flexible.

Further, it is sometimes impossible to find a solution that fully satisfies all rules of the program. In this case, one might still wish to look for a solution satisfying the program at least to a reasonably high degree. At other times, it may not even be required to obtain a solution that satisfies a program fully. That is, one might be more interested in a solution satisfying the program to a satisfactory high degree, especially if this solution comes at a lower cost. Consider the following problem based on an example from [2].

Example 1 There are four different kinds of sports that we like to practice to some degree. However, only certain combinations of sports lead to a full body exercise. Furthermore, some of the sports complement each other, i.e. less practice of one automatically leads to more practice of the other (rules $r_1 \dots r_4$ in the program below).

$$\begin{aligned}
r_1 &: \quad \mathit{lift_weights} \leftarrow \text{not } \mathit{swim} \\
r_2 &: \quad \mathit{swim} \leftarrow \text{not } \mathit{lift_weights} \\
r_3 &: \quad \mathit{run} \leftarrow \text{not } \mathit{play_ball} \\
r_4 &: \quad \mathit{play_ball} \leftarrow \text{not } \mathit{run} \\
r_5 &: \mathit{full_body_exercise} \leftarrow \mathit{lift_weights}, \mathit{run} \\
r_6 &: \mathit{full_body_exercise} \leftarrow \mathit{swim}, \mathit{play_ball} \\
r_7 &: \quad \leftarrow \text{not } \mathit{full_body_exercise}
\end{aligned}$$

The two classical answer sets of this program are

$$M = \{\mathit{full\_body\_exercise}, \mathit{lift\_weights}, \mathit{run}\}$$

and

$$N = \{\mathit{play\_ball}, \mathit{full\_body\_exercise}, \mathit{swim}\} \ .$$

Hence, to achieve a full body exercise, one needs to practice either weight lifting and running (M), or ball playing and swimming (N) to the highest degree. ASP only considers the alternatives of practicing a sport to the highest degree, or not practicing it at all. Likewise, ASP only considers either achieving a full body exercise, or none at all. However, in practice, we might be reluctant to carry out the complete sport programs necessary to achieve a full body exercise. Nevertheless we may want to put in *some* physical effort: say, we might be interested to know which combinations of the four sports we should practice, and to what degree, such that an acceptable degree, e.g. 0.7, of full body exercise is obtained.

Fuzzy logic is a suitable framework for dealing with degrees of truth and satisfaction [31]. In its most general form, fuzzy logic considers a complete lattice $\mathcal{L}$ of truth values on which it redefines the classical operations of negation, conjunction, disjunction and implication; in such a way that they correspond to the classical ones in the top and bottom elements of the lattice. One of the strengths of fuzzy logic regarding these operations is that a user can freely choose, depending on the type of application under consideration, which specific definition she uses for the operations.

A combination with fuzzy logic increases the flexibility and hence the application potential of ASP. Such flexibility can be introduced at several levels. In the fuzzy answer set programming (FASP) framework introduced in this paper, we consider fuzzy answer sets, which means that literals can belong to an answer set to a certain

extent, as opposed to either belonging to the answer set or not. In accordance, the literals in a program can be true to a certain degree. We also relax the definition of consistency to allow that, if desired, both a and $\neg a$ can be true to a certain degree at the same time without necessarily loosing consistency. Similarly, we allow for a more flexible interpretation of negation as failure. Crucial to our approach is the notion of a satisfaction function, as it enables us to compute the extent to which a rule is satisfied under a given fuzzy interpretation. The satisfaction function is then used to develop the concept of a fuzzy model. Next, we adapt the classical notion of an unfounded set to the setting of fuzzy interpretations, which allows us to bring to surface whether a fuzzy model is indeed supported by a program, in other words whether it deserves the name of fuzzy answer set.

The rest of the paper is organized as follows. In Section 2 we give some preliminaries on fuzzy logic and answer set programming, while we introduce the combination of both, i.e. fuzzy answer set programming (FASP), in Section 3. Section 4 shows how FASP can be combined with fuzzy input while the relationship between minimal fuzzy models and answer sets is discussed in Section 5. Before giving some comparison with related work in Section 7, we show in Section 6 how the classical answer set semantics can be retrieved from FASP. Finally, we conclude and give some directions for future research in Section 8.

2 Preliminaries

2.1 Truth lattices

In this paper, we consider a complete truth lattice, i.e. a partially ordered set $(\mathcal{L}, \leq_{\mathcal{L}})$ such that every subset of $\mathcal{L}$ has an infimum (greatest lower bound) and a supremum (least upper bound), which we denote by inf and sup respectively [4]. Such a lattice is often denoted by $\mathcal{L}$, tacitly assuming the ordering $\leq_{\mathcal{L}}$. Furthermore, we use $0_{\mathcal{L}}$ and $1_{\mathcal{L}}$ to denote respectively the smallest and the greatest element[2] of $\mathcal{L}$. If the lattice $\mathcal{L}$ under consideration is clear from the context, we omit the subscripts and use $\leq, 0$ and 1 to denote the ordering, the smallest and the greatest element respectively.

The traditional logical operations of negation, conjunction, disjunction, and implication can be generalized to logical operators acting on truth values of $\mathcal{L}$ (see e.g. [21]).

A *negator* on $\mathcal{L}$ is any $\mathcal{L} \to \mathcal{L}$ mapping $\mathcal{N}$ satisfying $\mathcal{N}(0) = 1$ and $\mathcal{N}(1) = 0$. Moreover we require $\mathcal{N}$ to be decreasing, i.e. $x_1 \leq x_2$ implies $\mathcal{N}(x_1) \geq \mathcal{N}(x_2)$ for all x_1 and x_2 in $\mathcal{L}$. A negator $\mathcal{N}$ is called involutive if $\mathcal{N}(\mathcal{N}(x)) = x$ for all x in $\mathcal{L}$.

A *triangular norm* $\mathcal{T}$ on $\mathcal{L}$ is any commutative and associative $\mathcal{L}^2 \to \mathcal{L}$ mapping $\mathcal{T}$ satisfying $\mathcal{T}(1, x) = x$, for all x in $\mathcal{L}$. Moreover we require $\mathcal{T}$ to be increasing in both of its components, i.e.[3] $x_1 \leq x_2$ implies $\mathcal{T}(x_1, y) \leq \mathcal{T}(x_2, y)$ for all x_1, x_2 and y in $\mathcal{L}$. A triangular norm, or t-norm for short, corresponds to conjunction.

A *triangular conorm* $\mathcal{S}$ on $\mathcal{L}$ is any increasing, commutative and associative $\mathcal{L}^2 \to \mathcal{L}$ mapping satisfying $\mathcal{S}(0, x) = x$, for all x in $\mathcal{L}$. Moreover we require $\mathcal{S}$ to

[2] In the literature one will also find the notation $\bot$ and $\top$ to denote $0_{\mathcal{L}}$ and $1_{\mathcal{L}}$ respectively.

[3] Note that the monotonicity of the second component immediately follows from that of the first component due to the commutativity.

be increasing in both of its components. A triangular conorm, t-conorm for short, corresponds to disjunction.

An *implicator* $\mathcal{I}$ on $\mathcal{L}$ is any $\mathcal{L}^2 \to \mathcal{L}$–mapping satisfying $\mathcal{I}(0,0) = 1$, and $\mathcal{I}(1,x) = x$, for all x in $\mathcal{L}$. Moreover we require $\mathcal{I}$ to be decreasing in its first, and increasing in its second component, i.e. $x_1 \leq x_2$ implies $\mathcal{I}(x_1, y) \geq \mathcal{I}(x_2, y)$ as well as $\mathcal{I}(y, x_1) \leq \mathcal{I}(y, x_2)$ for all x_1, x_2 and y in $\mathcal{L}$. Every implicator induces a negator by defining $\mathcal{N}(x) = \mathcal{I}(x, 0)$.

The dual of a t-norm $\mathcal{T}$ w.r.t. a negator $\mathcal{N}$ is a t-conorm $\mathcal{S}$ defined as $\mathcal{S}(x,y) = \mathcal{N}(\mathcal{T}(\mathcal{N}(x), \mathcal{N}(y)))$ for all x and y in $\mathcal{L}$. The mapping $\mathcal{I}_{\mathcal{S},\mathcal{N}}$ defined by $\mathcal{I}_{\mathcal{S},\mathcal{N}}(x,y) = \mathcal{S}(\mathcal{N}(x), y)$ is an implicator, usually called S-implicator (induced by $\mathcal{S}$ and $\mathcal{N}$). On the other hand, the mapping $\mathcal{I}_{\mathcal{T}}$ defined by $\mathcal{I}_{\mathcal{T}}(x,y) = \sup\{\lambda | \lambda \in \mathcal{L} \text{ and } \mathcal{T}(x,\lambda) \leq y\}$ is an implicator, usually called the residual implicator or R-implicator (of $\mathcal{T}$). If the partial mappings of $\mathcal{T}$ are supmorphisms,[4] then $\mathcal{T}$ and $\mathcal{I}_{\mathcal{T}}$ satisfy the residual principle or adjoint condition

$$\mathcal{T}(x,y) \leq z \text{ iff } x \leq \mathcal{I}_{\mathcal{T}}(y,z)$$

While the framework we will introduce to perform fuzzy answer set programming in Section 3 can be used in combination with any complete lattice, we will mostly restrict ourselves for the examples in the current paper to the complete lattice $([0,1], \leq)$. The following example presents some fuzzy logical operators on this lattice.

Example 2 The mapping $\mathcal{N}_s$ defined as $\mathcal{N}_s(x) = 1 - x$ for all x in $[0,1]$ is called the standard negator. The t-norms $\mathcal{T}_{\mathrm{M}}$, $\mathcal{T}_{\mathrm{P}}$, and $\mathcal{T}_{\mathrm{W}}$ and their dual t-conorms $\mathcal{S}_{\mathrm{M}}$, $\mathcal{S}_{\mathrm{P}}$, and $\mathcal{S}_{\mathrm{W}}$ w.r.t. the standard negator, are defined as

$$\begin{array}{lcllcl}
\mathcal{T}_{\mathrm{M}}(x,y) & = & \min(x,y) & \mathcal{S}_{\mathrm{M}}(x,y) & = & \max(x,y) \\
\mathcal{T}_{\mathrm{P}}(x,y) & = & x \cdot y & \mathcal{S}_{\mathrm{P}}(x,y) & = & x + y - x \cdot y \\
\mathcal{T}_{\mathrm{W}}(x,y) & = & \max(x + y - 1, 0) & \mathcal{S}_{\mathrm{W}}(x,y) & = & \min(x + y, 1)
\end{array}$$

for all x and y in $[0,1]$. They induce the following implicators (the mappings on the right are R-implicators while those on the left are S-implicators; for ease of notation the inducing negator $\mathcal{N}_s$ has been omitted):

$$\begin{array}{lcllcl}
\mathcal{I}_{\mathcal{S}_{\mathrm{M}}}(x,y) & = & \max(1-x, y) & \mathcal{I}_{\mathcal{T}_{\mathrm{M}}}(x,y) & = & \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{else} \end{cases} \\
\mathcal{I}_{\mathcal{S}_{\mathrm{P}}}(x,y) & = & 1 - x + x \cdot y & \mathcal{I}_{\mathcal{T}_{\mathrm{P}}}(x,y) & = & \begin{cases} 1, & \text{if } x \leq y \\ \frac{y}{x}, & \text{else} \end{cases} \\
\mathcal{I}_{\mathcal{S}_{\mathrm{W}}}(x,y) & = & \min(1 - x + y, 1) & \mathcal{I}_{\mathcal{T}_{\mathrm{W}}}(x,y) & = & \min(1 - x + y, 1)
\end{array}$$

Each of these t-norms and its associated R-implicator satisfy the residuation principle. The above mentioned S-implicators induce the standard negator $\mathcal{N}_s$, while $\mathcal{I}_{\mathcal{T}_{\mathrm{M}}}$ and $\mathcal{I}_{\mathcal{T}_{\mathrm{P}}}$ induce the Gödel negator[5] $\mathcal{N}_g$ defined by $\mathcal{N}_g(x) = 1$ if $x = 0$ and $\mathcal{N}_g(x) = 0$ otherwise.

[4] A $\mathcal{L} \to \mathcal{L}$ mapping g is called a supmorphism iff for each family $(x_i)_{i \in I}$ in $\mathcal{L}$ it holds that $\sup\{g(x_i) \mid i \in I\} = g(\sup\{x_i \mid i \in I\})$.

[5] This negator is also known in the literature as the Heyting negator.

A fuzzy set in U is a $U \mapsto \mathcal{L}$ mapping. The set of all elements that have a non-zero membership degree in a fuzzy set A in U is called the *support*, i.e. $\text{supp}(A) = \{x | x \in U \wedge A(x) \geq_{\mathcal{L}} 0_{\mathcal{L}}\}$. For fuzzy sets A and B in U, A is said to be included in B, denoted by $A \subseteq_{\mathcal{L}} B$, iff $A(u) \leq_{\mathcal{L}} B(u)$ for all u in U. As usual, we have $A \subset_{\mathcal{L}} B$ iff $A \subseteq_{\mathcal{L}} B$ and not $B \subseteq_{\mathcal{L}} A$. When the lattice under consideration is clear from the context, we will omit the subscripts.

2.2 Answer set programming

We give some preliminaries concerning the answer set semantics for logic programs [3]. A *literal* is an atom a or a negated atom $\neg a$. For a set of literals X, we use $\neg X$ to denote $\{\neg l \mid l \in X\}$ where $\neg\neg a = a$. When $X \cap \neg X = \emptyset$ we say that X is *consistent*. An *extended literal* is a literal or a *naf-literal* of the form not l where l is a literal. The latter form denotes negation as failure. For a set of extended literals Y, we use Y^- to denote the set of ordinary literals underlying the naf-literals in Y, i.e. $Y^- = \{l \mid \text{not } l \in Y\}$. Further, we use not X to denote the set $\{\text{not } l \mid l \in X\}$. An extended literal l is true w.r.t. X, denoted $X \models l$, if $l \in X$ in case l is ordinary, or $a \notin X$ if $l = \text{not } a$ for some ordinary literal a. As usual, $X \models Y$ iff $\forall l \in Y \cdot X \models l$.

A *rule* is of the form $a \leftarrow \beta$, where[6] a is either a literal or the symbol $\bot$ and β is a finite set of extended literals. To denote the head a of the rule, we use $H(a \leftarrow \beta)$, while $B(a \leftarrow \beta)$ is used to denote the body β. When the *head* of a rule r is the symbol $\bot$, i.e. $H(r) = \bot$, the rule is called a *constraint*, while rules with an empty body, i.e. $B(r) = \emptyset$ are called *facts*. For constraints, we normally omit the head symbol, i.e. we use $\leftarrow \beta$ instead of $\bot \leftarrow \beta$.

A finite set of rules is called a (*logic*) *program*. The *Herbrand base* $\mathcal{B}_P$ of a program P contains all atoms appearing in P. The set of all literals that can be formed with the atoms in P, denoted by Lit_P, is defined by $\text{Lit}_P = \mathcal{B}_P \cup \neg\mathcal{B}_P$. Similarly, we define the set of all extended literals that can be formed with the atoms in P as $\text{Elit}_P = \text{Lit}_P \cup \text{not Lit}_P$. Any consistent subset $I \subseteq \text{Lit}_P$ is called an *interpretation* of P.

A rule $r = a \leftarrow \beta$ is *satisfied* by an interpretation I, denoted $I \models r$, if $I \models a$, whenever $I \models \beta$, i.e. if r is *applicable* ($I \models \beta$), then it must be *applied* ($I \models \{a\} \cup \beta$). As $I \not\models \bot$, this implies that a constraint rule can only be satisfied if it is not applicable ($I \not\models \beta$). For a program P, an interpretation I is called a *model* of P if $\forall r \in P \cdot I \models r$, i.e. I satisfies all rules in P. It is a minimal model of P if there is no model J of P such that $J \subset I$.

A *simple program* is a program without negation as failure. For simple programs P, we define an *answer set* of P as a minimal model of P. On the other hand, for a program P, i.e. a program containing negation as failure, we define the *GL-reduct* [14] for P w.r.t. I, denoted P^I, as the program consisting of those rules[7] $a \leftarrow (\beta \setminus \text{not } \beta^-)$ where $a \leftarrow \beta$ is in P and $I \models \text{not } \beta^-$. Note that all rules in P^I are free from negation as failure, i.e. P^I is a simple program. An interpretation I is then an *answer set* of P iff I is a minimal model of the GL-reduct P^I.

[6] For simplicity, we assume that programs have already been grounded.

[7] As usual, $\setminus$ denotes set difference.

Example 3 Consider the program

$$r_1 : a \leftarrow \text{not } b \qquad r_2 : b \leftarrow \text{not } a$$

Clearly, both $\{a\}$ and $\{b\}$ are answer sets of this program as the GL-reducts $P^{\{a\}} = \{a \leftarrow\}$ and $P^{\{b\}} = \{b \leftarrow\}$ have $\{a\}$ and $\{b\}$ respectively as their minimal model. On the other hand, $\emptyset$ and $\{a, b\}$ are not answer sets. For the former interpretation, the reduct $P^{\emptyset} = \{a \leftarrow\ ;\ b \leftarrow\}$ has $\{a, b\}$ as its minimal model which differs from $\emptyset$, while the latter has an empty reduct, thus an empty minimal model, which differs from $\{a, b\}$.

An alternative characterization of answer sets can also be given in terms of unfounded sets [22, 26]. For a program P and an interpretation I, a set of literals $X \subseteq \text{Lit}_P$ is called an *unfounded set* w.r.t. I iff for each literal $l \in X$ and each rule $l \leftarrow \beta \in P$ we either have (1) $\beta \cap X \neq \emptyset$, or (2) $I \not\models \beta$. Intuitively, (1) covers the case of literals that depend on each other to motivate their presence in I, while (2) singles out rules that are not applicable, as they cannot motivate the presence of a literal in I. An interpretation I is then called *unfounded-free* iff $I \cap X = \emptyset$ for every unfounded set X w.r.t. I. Now, it can be shown that a model S is an answer set of P iff S is unfounded-free.

Example 4 Consider the program

$$r_1 : a \leftarrow \text{not } b \qquad r_2 : b \leftarrow \text{not } a \qquad r_3 : c \leftarrow c, a \qquad r_4 : c \leftarrow \text{not } a$$

One can see that $S = \{a, c\}$ is a model of the above program. However, $X = \{c\}$ is unfounded w.r.t. S as the body of r_3 contains a literal from X, and the body of r_4 is not applicable w.r.t. S, i.e. $S \not\models \text{not } a$. As a result, S is not unfounded free and thus not an answer set of the program.

On the other hand, one can check that $T = \{a\}$ is an answer set of the program as T is a model and clearly not unfounded. Indeed, the only rule with a in the head is r_1 which only contains naf-literals, i.e. condition (1) is not satisfied, and its body is satisfied w.r.t. T, i.e. condition (2) is not satisfied.

3 Fuzzy answer set programming

Classical ASP, as defined in the previous subsection, is in some ways a very strict framework in its semantics. In particular, an answer set is required to satisfy all rules of the program fully. In a more flexible setting, we wish to be able to deal with interpretations that satisfy rules possibly only to a certain extent. To this end, we allow literals to be true to a degree, as opposed to either being true or not true. As such, interpretations, and hence also answer sets, become fuzzy sets in Lit_P.

As the high configurability of fuzzy logic can be seen as one of its main strengths, we will adopt this behavior to the FASP framework presented in this section. Therefore, we allow a user to choose, in function of the application at hand, how the different classical operations need to be interpreted. More specifically, a user has to fix a complete lattice $\mathcal{L}$ first. Then, she has to choose two negators $\mathcal{N}_c$ and $\mathcal{N}_n$, which will be used to define consistency and the semantics of negation as failure respectively. Further, two t-norms $\mathcal{T}_c$ and $\mathcal{T}_a$ need to be fixed, respectively used

for defining consistency and applicability of rules. Also an implicator $\mathcal{I}$ is needed to obtain the degree of satisfaction of a rule. Finally, two aggregators $\mathcal{A}_c$ and $\mathcal{A}_p$ are needed, where $\mathcal{A}_c$ combines the consistency degrees of the different atoms and their negation into a consistency degree for the interpretation, while $\mathcal{A}_p$ combines all the degrees of satisfaction of rules into a single truth value denoting the degree in which a fuzzy interpretation is a fuzzy model. For the rest of this paper, we assume, without loss of generality, that the above choices have been made, and we will not repeat them everytime in the definitions, but just use them. Section 6 contains a discussion on choices of the fuzzy logical operators that behave more answer set like than others.

The first classical notions that need to be tackled are containment and consistency. In ASP a literal l is either true or false; and thus it is either contained in an interpretation or not. When both l and $\neg l$ are contained in an interpretation, it is said to be inconsistent. In a fuzzy context, an interpretation is a fuzzy set to allow for a literal l to be true to a degree. Furthermore, both l and $\neg l$ can be a bit true in a consistent way, making a modified notion of consistency necessary.

Definition 1 Let P be a program. A *fuzzy interpretation* I for P is a fuzzy set in Lit_P, i.e. a $I : \mathrm{Lit}_P \mapsto \mathcal{L}$ mapping. With I we associate a consistency function $I_c : \mathcal{B}_P \mapsto \mathcal{L}$, defined as $I_c(a) = \mathcal{N}_c(\mathcal{T}_c(I(a), I(\neg a)))$ for each $a \in \mathcal{B}_P$. Further, I is called x**-consistent**, $x \in \mathcal{L}$, iff $\mathcal{A}_c(\mathcal{B}_P, I_c) \geq x$.

Intuitively, the consistency function computes the degree of consistency of a particular atom and its negation in an interpretation, while the definition of x-consistency allows a user to choose how the individual consistencies of the atoms and their negations have to be combined into a degree of consistency for the whole interpretation. By choosing an appropriate lower bound x, the user can then fix the point where an interpretation is no longer considered consistent. Note that by using the aggregator $\mathcal{A}_c$ the user can choose to ignore certain inconsistencies, or she can allow certain literals to be more inconsistent than others. However, we demand that an aggregator is increasing whenever the degrees of the individual consistencies are increasing. The classical notion of an interpretation emerges from the above definition for the lattice $\mathcal{L} = \{0, 1\}$ by taking $\mathcal{A}_c(\mathcal{B}_P, I_c) = \inf\{I_c(a) \mid a \in \mathcal{B}_P\}$. In this particular case, an interpretation I is called 1-consistent iff there does not exist an a in $\mathcal{B}_P$ such that both $I(a) = 1$ and $I(\neg a) = 1$.

As fuzzy interpretations only assign truth values to ordinary literals explicitly, we need a mechanism to retrieve truth values for naf-literals. While complementary literals l and $\neg l$ are only weakly related to each other using $\mathcal{N}_c$, $\mathcal{T}_c$, $\mathcal{A}_c$ and a certain x-consistency boundary, naf-literals l and not l need a tighter connection since, intuitively, a naf-literal not l can only be true to the degree that the underlying ordinary literal l is false, and vice versa. Hence, we use $\mathcal{N}_n$ to extend a fuzzy interpretation I to cover naf-literals by defining $I(\mathrm{not}\, l) = \mathcal{N}_n(I(l))$ for each $l \in \mathrm{Lit}_P$.

Having fuzzy interpretations and x-consistency, we need to redefine the satisfaction of rules. While a rule in ASP is either satisfied or not, in a more flexible setting we should allow a rule to be partially (to a certain degree) satisfied. Further, each rule does not have to be satisfied to the same degree, which is, e.g., useful in applications having preferences among rules. To obtain these degrees, we use $\mathcal{T}_a$ and $\mathcal{I}$ to induce, for a fuzzy interpretation I, a satisfaction function $I_{\models}$ that assigns a truth value to the

bodies of rules and to the rules themselves. Later on, this satisfaction function will be used, in combination with the aggregator $\mathcal{A}_p$, to obtain the degree in which a fuzzy interpretation is a model of a program.

Definition 2 Let P be a program and let I be a fuzzy interpretation. The induced satisfaction function $I_{\models} : 2^{\text{Elit}_P} \cup P \mapsto \mathcal{L}$ is defined by

$$\begin{aligned} I_{\models}(\emptyset) &= 1 \\ I_{\models}(\{l\} \cup \beta) &= \mathcal{T}_a(I(l), I_{\models}(\beta)) \\ I_{\models}(\ \leftarrow \beta) &= \mathcal{I}(I_{\models}(\beta), 0) \\ I_{\models}(l \leftarrow \beta) &= \mathcal{I}(I_{\models}(\beta), I(l)) \end{aligned}$$

Note that $I_{\models}(\{l\}) = I(l)$ and $I_{\models}(\text{not } l) = \mathcal{N}_n(I(l))$, as $\{l\} = \{l\} \cup \emptyset$ and $\mathcal{T}_a(I(l), 1) = I(l)$. Intuitively, $I_{\models}(s)$, with $s \in P$, defines to which degree a rule s is satisfied taking into account the truth assignments of the head and body of s in I. To define a fuzzy model, the different $I_{\models}(s)$, with $s \in P$, need to be accumulated in some way. The user defined aggregator $\mathcal{A}_p$, which takes as input a program and a satisfaction function, will accomplish this job and result in a truth value denoting the degree in which the fuzzy interpretation I is a model of P. E.g., in an application working with different knowledge bases, where each knowledge base has a certain degree of trust, one could take a weighted aggregator into account. We demand that the aggregator is increasing whenever the degrees of satisfaction of the rules increase.

Definition 3 Let P be a program and let I be an x-consistent fuzzy interpretation. Then, I is an x-consistent *fuzzy y-model* of P, $y \in \mathcal{L}$, iff $\mathcal{A}_p(P, I_{\models}) \geq y$.

In the rest of the paper, a 1-consistent fuzzy y-model is also referred to as a fuzzy y-model, for short.

Example 5 Consider the lattice $\mathcal{L} = [0, 1]$ and the program

$$r_1 : a \leftarrow \text{not } b \qquad r_2 : b \leftarrow \text{not } a \qquad r_3 : c \leftarrow a$$

and consider the fuzzy interpretations[8] $K = \{(a, 0.9), (b, 0.3), (c, 0.2)\}$ and $L = \{(a, 0.4), (b, 0.7), (c, 0.8)\}$. Both of these fuzzy interpretations are 1-consistent, taking $\mathcal{A}_c(\mathcal{B}_P, I_c) = \inf\{I_c(a) \mid a \in \mathcal{B}_P\}$ and independently of the choices for $\mathcal{N}_c$ and $\mathcal{T}_c$. For negation as failure, we use the negator $\mathcal{N}_s$. To compute the satisfaction of the rules, we use the implicator $\mathcal{I}_{\mathcal{S}_M}$. Finally, as an aggregator for the rules, we use $\mathcal{A}_p(P, I_{\models}) = \inf\{I_{\models}(s) \mid s \in P\}$, i.e. the weakest rule dominates the solution.

Now, one can check that we have

$$\begin{aligned} K_{\models}(r_1) &= \max(1 - K(\text{not } b), K(a)) = \max(1 - \mathcal{N}_s(K(b)), K(a)) \\ &= \max(1 - (1 - 0.3), 0.9) = 0.9 \end{aligned}$$

[8] As usual, a fuzzy set I in Lit_P is denoted as $\{(l, x) \mid I(l) = x \wedge l \in \text{Lit}_P\}$, omitting the literals $(l, 0)$.

Similarly, we have $K_{\models}(r_2) = \max(1 - (1 - 0.9), 0.3) = 0.9$ and $K_{\models}(r_3) = \max(1 - 0.9, 0.2) = 0.2$. As a result, K is a fuzzy 0.2-model of P. On the other hand, one can verify that $L_{\models}(r_1) = L_{\models}(r_2) = 0.7$ and $L_{\models}(r_3) = 0.8$, yielding that L is a fuzzy 0.7-model of P.

Example 6 In Example 5, the weakest rule dominates the outcome. The inf operator used in Example 5 for aggregating the satisfaction of the rules is the fuzzy logical counterpart of the universal quantifier, expressing that *all* rules need to be satisfied (more accurately, the program can only be a fuzzy model to the degree to which *all* rules are satisfied). A more flexible approach can aim for satisfaction of *most* of the rules. To model the vague quantifier "most," we use an ordered weighted averaging (OWA) operator [30]. Such an aggregator is defined by means of a weighting vector $(w_1, w_2, \ldots, w_n)$. All weights are positive and they sum up to 1. The corresponding aggregator for rules is given by

$$\mathcal{A}_p(P, I_{\models}) = \sum_{s_j \in P} w_j \cdot I_{\models}(s_j)$$

with $I_{\models}(s_j)$ being the jth largest satisfaction value of all rules in P. A key feature of an OWA operator is thus the ordering of the rules according to their satisfaction values prior to calculating the weighted average.

Going back to Example 5, we can use the weighting vector $(0.5, 0.5, 0.0)$, expressing that 2 out of the 3 rules of the program need to be satisfied. Aggregating the values $K_{\models}(r_1) = 0.9$, $K_{\models}(r_2) = 0.9$, and $K_{\models}(r_3) = 0.2$, we obtain $0.5 \cdot 0.9 + 0.5 \cdot 0.9 + 0.0 \cdot 0.2 = 0.9$, in other words K is a fuzzy 0.9-model of P. Similarly we aggregate $L_{\models}(r_3) = 0.8$, $L_{\models}(r_1) = 0.7$, and $L_{\models}(r_2) = 0.7$ as $0.5 \cdot 0.8 + 0.5 \cdot 0.7 + 0.0 \cdot 0.7$, hence L is a fuzzy 0.75-model of P. In both cases we ignored the rule that is least satisfied in computing the overall satisfaction. As a result, one weak rule can no longer dominate the solution. In contrast with the outcome in Example 5, K is now considered to be a slightly better fuzzy model than L because it satisfies most of the rules to a slightly higher degree.

The above definitions are conservative extensions of classical principles, i.e. the classical definitions are special cases of the ones presented here. Hence it is not surprising that the extensions suffer the same difficulties as the classical ones when used to define a fuzzy answer set semantics. The first problem encountered is illustrated by the following example.

Example 7 Consider the program

$$r_1 : a \leftarrow b \qquad r_2 : b \leftarrow a$$

Both $I = \{(a, 0), (b, 0)\}$ and $J = \{(a, 1), (b, 1)\}$ are "perfect" fuzzy models of the program as they both satisfy all rules to a maximal degree 1. However, in traditional ASP, the set $I' = \{a, b\}$ is unfounded [using condition (1) of the unfounded set definition], making I' not an answer set of the program.

To solve the above problem, we could adopt condition (1) of unfounded sets to a fuzzy version of unfounded sets. However, we also need a fuzzy adaption for condition (2) of the unfounded set definition, as witnessed by the following example.

Example 8 Consider the program

$$r_1 : a \leftarrow \qquad r_2 : b \leftarrow a, \text{not } c \qquad r_3 : d \leftarrow \text{not } b$$

and the fuzzy interpretations $K = \{(a, 0.9), (b, 0.2), (c, 0.9), (d, 0.9)\}$ and $L = \{(a, 0.9), (b, 0.9), (d, 0.1)\}$. For negation as failure, we use the negator $\mathcal{N}_s$. To evaluate the body of r_2 we use $\mathcal{T}_a = \mathcal{T}_{\mathrm{M}}$, while the implicator that we use is $\mathcal{I} = \mathcal{I}_{\mathcal{T}_{\mathrm{M}}}$. Further, we use again $\mathcal{A}_p(P, I_{\models}) = \inf\{I_{\models}(s) \mid s \in P\}$. One can verify that K and L are both fuzzy 0.9-models. However, intuitively, K seems less acceptable than L. E.g., when the rule r_2 is considered w.r.t. K, we see that $K(b) = 0.2$, while $K(b) = 0.1$ would suffice to obtain the same degree of satisfaction for r_2, i.e. $K_{\models}(r_2) = \mathcal{I}(\mathcal{T}_a(0.9, 1 - 0.9), 0.2) = \mathcal{I}(0.1, 0.2) = 1 = \mathcal{I}(0.1, 0.1)$. Further, there is no support for accepting c at degree 0.9, as there is no applicable rule with c in the head.

On the other hand, L does not suffer from these problems as each literal in L appears in the head of a rule and none of the truth degrees of the head literals can be lowered without lowering the degree of satisfaction of the corresponding rule.

In traditional ASP, the above problem is solved by taking the GL-reduct which will remove, for $I = \{a, b, c, d\}$, the rule r_3 from the reduct P^I, because r_3 is not applicable due to the not b literal in its body. Now, the minimal model of this reduct does not equal I, hence, it is rejected as an answer set. Note that the removal of a rule does not mean that this rule does not have to be satisfied anymore. On the contrary, in the example above, rule r_3 is removed because it is not applicable under interpretation I, hence it is satisfied by default, independently of the truth value of d. Therefore, the above case is handled by condition (2) in the unfounded set characterization of answer sets.

Note that in traditional ASP, there are two possible scenarios for a model I to satisfy a rule of the form $l \leftarrow \beta$. Either it is applicable ($I \models \beta$), hence l must assume the truth value 1 to satisfy the rule, or it is unapplicable ($I \not\models \beta$), hence the rule is satisfied by default and l can assume any truth value from the lattice $\mathcal{L} = \{0, 1\}$. In the first case, the truth value of l is fully determined by the rule, while in the latter case, the rule does not impose any restrictions on the truth value of l, hence taking it into account does not influence the result and we can remove the rule.

In FASP, such a removal strategy for naf-literals is not feasible as such literals may be true only to a certain degree, making the bodies of some rules applicable to a certain degree, which requires that they also need to be applied to a certain degree. Hence, as opposed to either fully determining the truth value of the head of a rule (full information), or leaving it completely arbitrary (no information), in FASP a rule may also carry *some* information that delimits the set of possible truth values that can be assumed by the head.

Thus, in a fuzzy adaption of condition (2) of unfounded sets, we will need to define for each rule r in the program a subset Y of $\mathcal{L}$ such that assigning any value

from Y to the head of r does not lower r's degree of satisfaction. However, when the lower values in the range are chosen, the rule is said to *support* its head literal better. Formally, the support of a rule $l \leftarrow \beta \in P$ w.r.t. a fuzzy interpretation I is defined as

$$I_s(l \leftarrow \beta) = \inf\{y \mid \mathcal{I}(I_{\models}(\beta), y) \geq I_{\models}(l \leftarrow \beta)\} \ .$$

Combining the above allows us to define a version of unfounded sets for fuzzy models.

Definition 4 Let P be a program and let I be a fuzzy interpretation. A set of literals X is an *unfounded set* w.r.t. I iff for each literal $l \in X$ and each rule $l \leftarrow \beta \in P$:

1. $\beta \cap X \neq \emptyset$; or
2. $I(l) > I_s(l \leftarrow \beta)$; or
3. $I_{\models}(\beta) = 0$ holds.

An interpretation I is *unfounded-free* iff $\text{supp}(I) \cap X = \emptyset$ for every unfounded set X w.r.t. I. An x-consistent unfounded-free fuzzy y-model of P is called an x-consistent *fuzzy y-answer set* of P.

In the rest of the paper, a 1-consistent fuzzy y-answer set is also referred to as a fuzzy y-answer set, for short.

Example 9 Reconsider Example 7. Clearly, $\text{supp}(J)$ is unfounded w.r.t. J as both r_1 and r_2 satisfy condition (1) of Definition 4. Obviously, I is unfounded-free as $\text{supp}(I) = \emptyset$, yielding that I is a fuzzy 1-answer set.

Example 10 Reconsider Example 8. One can check that $X = \{b, c, d\}$ is an unfounded set w.r.t. K, as

- For $b \in X$, we have $K_s(r_2) = \inf\{y \mid \mathcal{I}(0.1, y) \geq 1\} = \inf[0.1, 1]$, yielding that $K(b) = 0.2 > K_s(r_2) = 0.1$;
- For $c \in X$, we have no rules in P with c in the head of the rule, vacuously satisfying the conditions in Definition 4; and
- For $d \in X$, we have $K_{\models}(r_3) = \mathcal{I}(1 - 0.2, 0.9) = \mathcal{I}(0.8, 0.9) = 1$ and thus $K_s(r_3) = \inf\{y \mid \mathcal{I}(0.8, y) \geq 1\} = \inf[0.8, 1]$, yielding that $K(d) = 0.9 > K_s(r_3) = 0.8$.

On the other hand, for L and the rules in P, we have

$$L(a) = \inf\{y \mid \mathcal{I}(1, y) \geq 0.9\} = \inf[0.9, 1]$$
$$L(b) = \inf\{y \mid \mathcal{I}(0.9, y) \geq 1\} = \inf[0.9, 1]$$
$$L(d) = \inf\{y \mid \mathcal{I}(0.1, y) \geq 1\} = \inf[0.1, 1]$$

As a result, L is clearly unfounded-free and thus a fuzzy 0.9-answer set of P.

The next example illustrates the need for condition (3) in the definition of an unfounded set. Without this condition, the fuzzy version would not behave similar to the classical one.

Example 11 Consider the single rule $a \leftarrow b$ and the fuzzy interpretation $I = \{(a, 0), (b, 0)\}$. Without the third condition, one can check that the set $X = \{a\}$ is not unfounded w.r.t. I. However, in the classical setting X is unfounded w.r.t. the interpretation $\emptyset$, as the only rule with the literal in the head is not applicable. The third condition handles this particular situation, i.e. when a rule is neither applicable, nor applied.

Note that our definition of unfounded set also correctly handles cases where a rule r has a low applicability w.r.t. a fuzzy interpretation I and because of that only motivates the head to the degree $0_{\mathcal{L}}$, i.e. $I(H(r)) = 0$, $I_s(r) = 0$ and $I_{\models}(B(r)) > 0$. In this setting, the head literal $H(r)$ of such a rule should not be a member of an unfounded set and our definition behaves like this as neither condition (2) nor condition (3) will be satisfied by such rules.

The following is a straightforward property, but useful in the context of computing fuzzy answer sets.

Proposition 1 *Let P be a program and let M be an x-consistent fuzzy y-answer set of P. Then, M is an x-consistent fuzzy z-answer set of P, with $z \leq y$.*

Proof Clearly, it follows from Definition 3 that a fuzzy y-model is a fuzzy z-model whenever $z \leq y$. Further, M being unfounded (Definition 4) does not depend on M being a fuzzy y-model, and thus the result follows. □

Intuitively, this implies that one can demand a lower bound z for the quality of the fuzzy answer set, which can lead to three possible scenarios during fuzzy answer set computation: (1) we get a solution of quality z, the minimum we demanded; or (2) we get a solution of quality $y > z$, i.e. for the price of computing a solution of quality z, we get one of a higher quality y; or (3) we get a solution of quality $x < z$, i.e. we need to backtrack and find a better solution.

Example 12 Reconsider Example 1 from the introduction. We are interested to know to what degrees we have to practice the various sports such that an acceptable degree, e.g. 0.7, of full body exercise is obtained. Since our main concern is a satisfactory degree of full body exercise, we will use an aggregator that gives more importance to the constraint rule r_7. An appropriate choice could be an aggregator that only takes r_7 into account. In this case a fuzzy interpretation is a model to the degree to which it satisfies r_7. Of course we also require the model to be unfounded-free, which is where other rules come into play.

Further, we will also use $\mathcal{T}_a = \mathcal{T}_{\mathrm{M}}$ to evaluate the body of rules, and the implicator $\mathcal{I} = \mathcal{I}_{\mathcal{T}_W}$ to evaluate the satisfaction of the rules. A fuzzy 0.7-answer set K for the above program must at least satisfy $K_{\models}(r_7) \geq 0.7$. This yields that

$$\min(1 - K(\text{not } \mathit{full\_body\_exercise}) + 0, 1) \geq 0.7 \ ,$$

which implies that $K(\text{not } \mathit{full\_body\_exercise}) \leq 0.3$, and thus, using $\mathcal{N}_s$ for negation as failure, that $K(\mathit{full\_body\_exercise}) \geq 0.7$. To have support for the literal, i.e. to

avoid that *full_body_exercise* is part of an unfounded set, one of the two rules r_5 or r_6 have to be made applicable to a certain degree in such a way that not both satisfy condition (2) of Definition 4, i.e.

$$0.7 \leq K(\mathit{full\_body\_exercise}) = \inf\{y \mid \mathcal{I}(K_{\models}(B(r_5)), y) \geq K_{\models}(r_5)\} \text{ , or}$$
$$0.7 \leq K(\mathit{full\_body\_exercise}) = \inf\{y \mid \mathcal{I}(K_{\models}(B(r_6)), y) \geq K_{\models}(r_6)\}$$

must hold. In turn this implies that some of the four sports will have to be exercised in a higher degree than others to achieve that sufficient degree of applicability of r_5 or r_6.[9]

One can verify that

$$K = \{(\mathit{lift\_weights}, 0.8), (\mathit{swim}, 0.2), (\mathit{run}, 0.7), (\mathit{play\_ball}, 0.3), (\mathit{full\_body\_exercise}, 0.7)\} \ ,$$

is a fuzzy 0.7-answer set of the above program.

Intuitively, this solution is acceptable as it describes a configuration where two sports, which are together in rule r_5, are assigned a higher degree than their complementary variants, and due to this choice we have support for full body exercise up to a degree of 0.7.

Similarly, it can be checked that

$$L = \{(\mathit{lift\_weights}, 0.9), (\mathit{swim}, 0.1), (\mathit{run}, 0.8), (\mathit{play\_ball}, 0.2), (\mathit{full\_body\_exercise}, 0.8)\} \ ,$$

is a fuzzy 0.7-answer set of the program, and even a 0.8-answer set.

On the other hand, one can check that for the fuzzy interpretation

$$M = \{(\mathit{lift\_weights}, 0.8), (\mathit{swim}, 0.2), (\mathit{run}, 0.3), (\mathit{play\_ball}, 0.7), (\mathit{full\_body\_exercise}, 0.7)\} \ ,$$

it turns out that $\{\mathit{full\_body\_exercise}\}$ is an unfounded set w.r.t. M. Indeed, for the rule r_5 we have

$$\inf\{y \mid \min(1 - 0.3 + y, y) \geq 1\} = \inf[0.3, 1] = 0.3 \neq 0.7 \ ,$$

while for the rule r_6 we have

$$\inf\{y \mid \min(1 - 0.2 + y, y) \geq 1\} = \inf[0.2, 1] = 0.2 \neq 0.7 \ .$$

Hence, M is not a fuzzy 0.7-answer set of the program, fitting our intuition.

Finally, the following example illustrates the usefulness of our semantics in cases where the classical answer set semantics fails to provide solutions.

[9] Note that this example also illustrates how the proposed framework can be used to do fuzzy diagnostic reasoning. The constraint r_7 can be seen as an encoding of the observations, r_5 and r_6 represent the system description, while r_1, r_2, r_3 and r_4 provide the explanations.

Example 13 Consider the program

$$r_1 : a \leftarrow \text{not } b \qquad r_2 : b \leftarrow \text{not } c \qquad r_3 : c \leftarrow \text{not } a$$

which clearly has no classical answer sets.

Now, take $\mathcal{N}_n = \mathcal{N}_s$, $\mathcal{I} = \mathcal{I}_{\mathcal{S}_\mathrm{P}}$ and $\mathcal{A}_p(P, I_\models) = \inf\{I_\models(s) \mid s \in P\}$. One can check that this program has no fuzzy 1-answer sets. The only possibility satisfying all rules to degree 1 would be {(a,1),(b,1),(c,1)}, which is clearly unfounded because of condition (3) in Definition 4.

However, if we relax to e.g. fuzzy 0.9-answer sets, we do get solutions. One such a solution is $I = \{(a, 0.8), (b, 0.8), (c, 0.8)\}$, where each rule is satisfied to the degree $\mathcal{I}_{\mathcal{S}_\mathrm{P}}(0.2, 0.8) = 1 - 0.2 + 0.16 = 0.96$. Another solution is $J = \{(a, 0.9), (b, 0.7), (c, 0.8)\}$, yielding that $J_\models(r_1) = 0.97$, $J_\models(r_2) = 0.94$ and $J_\models(r_3) = 0.98$. On the other hand, going too low with the truth values for a, b and c will no longer yield fuzzy 0.9-answer sets. E.g., $K = \{(a, 0.3), (b, 0.3), (c, 0.3)\}$ is only a fuzzy 0.51-answer set of the program.

Intuitively, the latter result is correct as the lower you go with the truth values, the higher the applicability of a rule becomes, and thus the lower the degree of rule satisfaction.

4 Fuzzy answer set programming with fuzzy input

In this section, we show how the fuzzy answer set semantics can be used in cases where we want to combine a fuzzy program with a fuzzy input set of literals. In classical answer set programming, one will construct for a program P and a set of input literals I, a program $\Pi(P, I)$ whose answer sets correspond to the solutions of the program enriched with the input. In the classical case, the program $\Pi(P, I)$ is defined as

$$\Pi(P, I) = P \cup \{l \leftarrow \mid l \in I\} \ .$$

Example 14 Consider the program containing the rules

$$\begin{aligned} \mathit{sell}(S) &\leftarrow \mathit{sell\_advice}(S, A) \\ \mathit{buy\_share}(S) &\leftarrow \mathit{buy\_advice}(S, A1), \mathit{buy\_advice}(S, A2), A1 \neq A2, \text{not } \mathit{sell}(S) \end{aligned}$$

which intuitively encodes that we buy a share on the stock market if we have two different buying advices for the share and there is no selling advice that we are aware of.[10] To avoid the long predicate names, we will use, in what follows, s, bs, ba and sa to denote *sell*, *buy_share*, *buy_advice* and *sell_advice* respectively.

[10]Note that the rule $\mathit{sell}(S) \leftarrow \mathit{sell\_advice}(S, A)$ does not mean we sell a share from the moment we have a single sell advice, but it is introduced to have a correct interpretation of the second rule. If we would have written the second rule as

$$\mathit{buy\_share(S)} \quad \leftarrow \quad \mathit{buy\_advice(S,A1)}, \mathit{buy\_advice(S,A2)}, A1 \neq A2, \text{not } \mathit{sell\_advice}(S, A3)$$

the program would have a totally different meaning (due to the process of grounding), i.e. we buy a share with two buy advices and if there is a source which has no selling advice.

Suppose we get the following input from the outside world, i.e.

$$I = \{ba(c1, a1), ba(c1, a2), ba(c2, a2), ba(c2, a3), ba(c3, a4), sa(c2, a4)\} \ .$$

One can easily verify that $\Pi(P, I)$ has a single answer set

$$S = I \cup \{bs(c1), s(c2)\} \ ,$$

which corresponds with our intuition.

However, in a fuzzy context the above input I will be more like

$$I^f = \{(ba(c1, a1), 0.8), (ba(c1, a2), 0.7), (ba(c2, a2), 0.9),$$
$$(ba(c2, a3), 0.6), (ba(c3, a4), 0.3), (sa(c2, a4), 0.7)\} \ .$$

Intuitively, this means for example that adviser $a1$ advises strongly, i.e. to degree 0.8, to buy the share $c1$, while there is only a weak advice from source $a4$ to buy $c3$, i.e. to degree 0.3.

To handle this kind of input, we can still use the construction $\Pi(P, I^f)$. To be formally correct, its definition is slightly different, i.e. $\Pi(P, I^f) = P \cup \{l \leftarrow \mid l \in \text{supp}(I^f)\}$. Further, we will assume, without loss of generality, that for a given fuzzy input I^f, there is no rule $a \leftarrow \beta \in P$ such that $a \in \text{supp}(I^f)$, i.e. none of the input literals appears in the head of the rules in P.

To retrieve intuitively correct solutions for $\Pi(P, I^f)$, one has to use an aggregator that satisfies an additional constraint, i.e. an aggregator $\mathcal{A}_p$ for $\Pi(P, I^f)$ has to be defined such that[11] $\mathcal{A}_p(\Pi(P, I^f), S) = 0$ if there exists an $l \in \text{supp}(I^f)$ for which $S_{\models}(l \leftarrow) \neq I^f(l)$. Intuitively, as an implicator always satisfies the condition $\mathcal{I}(1, x) = x$ for every $x \in \mathcal{L}$ and because the input literals do not appear in the head of the rules in P, such an aggregator implies that any fuzzy y-model S, with $y > 0$, will always contain the fuzzy input I^f.

Example 15 Reconsider the program from Example 14 and the fuzzy input I^f we defined before. The program $\Pi(P, I^f)$ contains, besides the rules $\{l \leftarrow \mid l \in \text{supp}(I^f)\}$ for the input, the rules[12]

$$r_1 : s(c2) \leftarrow sa(c2, a4)$$
$$r_2 : bs(c1) \leftarrow ba(c1, a1), ba(c1, a2), \text{not } s(c1)$$
$$r_3 : bs(c2) \leftarrow ba(c2, a2), ba(c2, a3), \text{not } s(c2)$$

In what follows, we will use $\mathcal{T}_a = \mathcal{T}_M$ to evaluate the bodies of the rules, i.e. the weakest literal in the body decides the degree of applicability. Further, we will use the implicator $\mathcal{I} = \mathcal{I}_{\mathcal{T}_M}$ and we will use an aggregator that computes the infimum of the satisfaction degrees of the rules in P, but which results in 0 if there is an

[11] Note that we allow the aggregator to be not increasing for input fact rules. Indeed, to guarantee that the answer set contains each input fact exactly to the degree specified in the fuzzy input, the aggregated value of the rule satisfactions drops to 0 as soon as the specified degree is surpassed in one (or more) of the input fact rules.

[12] Note that we only provide the grounded rules that are useful, as the full grounding would make the program unnecessarily large.

$l \in \text{supp}(I^f)$ for which $S_{\models}(l \leftarrow) \neq I^f(l)$. Finally, we will restrict ourselves to fuzzy 1-answer sets in this example, i.e. all the rules in P have to be fully satisfied, and drop the "1-" qualification accordingly.

Clearly, as we have no selling advices for the share $c1$, any fuzzy answer set S will always yield $S(s(c1)) = 0$. Further, for $c2$, we will have, due to the rule r_1, that $S(s(c2)) = S(sa(c2, a4)) = 0.7$. If not, then $S(s(c2)) < 0.7$ would yield that $S_{\models}(r_1) < 1$ making S not a 1-answer set. On the other hand, if $S(s(c2)) > 0.7$, then $\{s(c2)\}$ is an unfounded set w.r.t. S as $\inf\{y \mid \mathcal{I}(0.7, y) \geq 1\} = \inf[0.7, 1] = 0.7$.

To evaluate the rule r_2, first note that, using $\mathcal{N}_s$ for negation as failure, $S(s(c1)) = 0$ implies $S(\text{not } s(c1)) = 1$. Applying $\mathcal{T}_a$ on the body of r_2 yields $\min\{0.8, 0.7, 1\} = 0.7$, which in turn implies that $S(bs(c1)) = 0.7$. Similarly, we have for the rule r_3 that $S(s(c2)) = 0.7$ implies that $S(\text{not } s(c2)) = 1 - 0.7 = 0.3$. This time, applying $\mathcal{T}_a$ on the body of r_3 yields $\min\{0.9, 0.6, 0.3\} = 0.3$.

As a result, $S = I^f \cup \{(s(c2), 0.7), (bs(c1), 0.7), (bs(c2), 0.3)\}$ is a fuzzy 1-answer set for P and input I^f.

The above example illustrates the usefulness of fuzzy answer set programming in comparison with classical answer set programming. While in the classical case we have either $bs(x)$ or not $bs(x)$ for each share x under consideration, the fuzzy case results in $bs(x)$ for each share, but with an additional degree indicating to what extent $bs(x)$ holds. E.g., if we lower the degree of the selling advice to 0.2 in the previous example, one can check that $S = I^f \cup \{(s(c2), 0.2), (bs(c1), 0.7), (bs(c2), 0.6)\}$ is also a fuzzy answer set, which is intuitively acceptable as we have two relatively strong buying advices for $c2$ and only a very weak selling advice.

5 Minimal fuzzy models and answer sets

From Proposition 1, it is clear that fuzzy answer sets are not truth minimal. However, if one wants such an additional minimization step, and not necessarily truth minimal, that can be accomplished in the usual way, e.g. as is done in the context of preferences in answer set programming [28].

Definition 5 Let P be a program and let $\preceq$ be a partial order on the fuzzy interpretations of P. An x-consistent fuzzy y-answer set M of P is $\prec$-*minimal* (or $\prec$-*preferred*) iff M is minimal w.r.t.[13] $\prec$ among the x-consistent fuzzy y-answer sets of P.

Example 16 Consider the simple single rule program $a \leftarrow$. Clearly, $\{(a, 0.7)\}$, $\{(a, 0.8)\}$ and $\{(a, 0.9)\}$ are fuzzy 0.7-answer sets of this program. However, when we consider $\subseteq$ and apply it to the above fuzzy 0.7-answer sets, we get

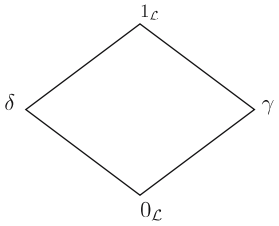
$$\{(a, 0.7)\} \subset \{(a, 0.8)\} \subset \{(a, 0.9)\} \ ,$$

yielding that $\{(a, 0.7)\}$ is a $\subset$-minimal fuzzy 0.7-answer set of this simple program.

[13] As usual we have $x \prec y$ iff $x \preceq y$ and not $y \preceq x$.

In the classical answer set semantics, we have for simple programs, i.e. programs without negation as failure, that minimal, w.r.t. subset inclusion, models coincide with answer sets. In the fuzzy answer set semantics, such a result does not hold in general for $\subset$-minimality, as witnessed by the following example.

Example 17 Consider the lattice $\mathcal{L}_f$ defined as $0 < \gamma < 1$ and $0 < \delta < 1$, which can be graphically represented as



Now, consider the simple program

$$r_1 : a \leftarrow \qquad r_2 : b \leftarrow a$$

and the implicator I_z defined as

$$I_z(x, y) = \begin{cases} \delta & \text{if } x = \gamma \wedge y = 0 \\ \gamma & \text{if } x = \delta \wedge y = 0 \\ y & \text{if } x = 1 \\ 1 & \text{otherwise} \end{cases}$$

One can check[14] that for this implicator the fuzzy interpretation $I = \{(a, \delta), (b, \delta)\}$ is a $\subset$-minimal 1-consistent fuzzy δ-model. If we lower the truth of a to 0, the rule satisfaction of r_1 also lowers to 0, making $\{(b, \delta)\}$ or $\emptyset$ at most fuzzy 0-models. Similarly, the fuzzy interpretation $\{(a, \delta)\}$ is only a fuzzy 0-model as $I_z(\delta, 0) = \gamma$ and $\inf(\{I_z(r_1), I_z(r_2)\}) = \inf(\{\gamma, \delta\}) = 0$.

However, one can check that I is not unfounded-free, as $\{b\}$ is an unfounded set w.r.t. I. Indeed, we have that

$$I_{\models}(r_2) = I_z(\delta, \delta) = 1 \ ,$$

and thus we get for b

$$\inf\{y \mid I_z(\delta, y) \geq I\} = \inf\{\delta, \gamma, I\} = 0 \neq I(b) = \delta \ .$$

As a result, I is not a 1-consistent fuzzy δ-answer set of P.

Intuitively, such problematic cases regarding minimality can happen in two cases, i.e. when we have non-total lattices and when the inverse use of the implicator yields a set of truth values that does not contain its own infimum. However, when we put a restriction on the type of lattices and on the implicator used, we can avoid the problems and show that $\subset$-minimal models are fuzzy answer sets.

[14] As the example is so simple, we do not need to specify any fuzzy operations for evaluating the body of rules or for defining consistency, as we only use the global conditions each negator or t-norm have to satisfy. For the aggregation of the rules, we will use inf.

In what follows, we assume that programs are simple, the lattice $\mathcal{L}$ is total and the implicator is a residual implicator that satisfies the residual principle or adjoint condition.

A mapping $f : P \mapsto \mathcal{L}$ is called an y-rule interpretation, $y \in \mathcal{L}$, for $\mathcal{A}$ iff $\mathcal{A}(P, f) \geq y$. For an y-rule interpretation f, a fuzzy interpretation I is called an f-rule model iff $I_{\models}(r) \geq f(r)$ for all $r \in P$. Due to the increasingness of the aggregator $\mathcal{A}$, an f-rule model is a fuzzy y-model.

The set of all $\subset$-minimal y-rule interpretations for $\mathcal{A}$ and P is denoted by $\mathcal{R}^y_{\mathcal{A}}$. The intuition is that each $f \in \mathcal{R}^y_{\mathcal{A}}$ assigns truth values to the rules such that if these values are used as satisfaction of the rules, the resulting solution must be a y-model and none of the satisfactions may drop or we cannot have a y-model anymore.

For an y-rule interpretation f and a fuzzy interpretation I, we define the f-support of a rule $r \in P$ w.r.t. I as

$$I_s(f, r) = \inf\{y \mid \mathcal{I}(I_{\models}(B(r)), y) \geq f(r)\} \ .$$

The support of a rule w.r.t. I is retrieved by choosing the satisfaction function $I_{\models}$ for f, i.e. $I_s(r) = I_s(I_{\models}, r)$. The following lemma holds due to the monotonicity of the fuzzy logical operators.

Lemma 1 *Let f be an y-rule interpretation, and let I and J be fuzzy interpretations. If $I \subseteq J$ then for every rule $r \in P$ it holds that $J_s(f, r) \geq I_s(f, r)$.*

In what follows, P_l is used, $l \in \mathrm{Lit}_P$, to denote the set $\{r \in P \mid H(r) = l\}$.

Definition 6 Let f be an y-rule interpretation, the *immediate consequence operator* $T_{P,f}$ is defined on a fuzzy interpretation I by, for any literal $l \in \mathrm{Lit}_P$,

$$T_{P,f}(I)(l) = \sup\{I_s(f, r) \mid r \in P_l\} \ . \tag{1}$$

The following is immediate from Definition 6 and Lemma 1.

Proposition 2 *The $T_{P,f}$ operator is monotonic, i.e. $T_{P,f}(I) \subseteq T_{P,f}(J)$ whenever $I \subseteq J$.*

Because of the restriction we placed on the implicator, we could also define our immediate consequence operator using the t-norm $\mathcal{T}$ underlying $\mathcal{I}$, as is done in the immediate consequence operator defined in [9], i.e.

$$T_{P,f}(I)(l) = \sup\{\mathcal{T}(f(r), I_{\models}(B(r))) \mid r \in P_l\} \ .$$

The following technical lemma will be useful later on. It says that $T_{P,f}(I)(a)$ is the minimal choice for a that is needed to increase I to an f-rule model on P_a.

Lemma 2 *Let I be a fuzzy interpretation and let f be an y-rule interpretation. Then*

$$\forall a \in \mathcal{B}_P \cup \neg\mathcal{B}_P \cdot \forall z < T_{P,f}(I)(a) \cdot \exists r \in P_a \cdot \mathcal{I}(I_{\models}(B(r)), z) < f(r)$$

and

$$\forall a \in \mathcal{B}_P \cup \neg\mathcal{B}_P \cdot \forall r \in P_a \cdot \mathcal{I}(I_{\models}(B(r)), T_{P,f}(I)(a)) \geq f(r) \ .$$

In what follows, we will use the notation $T^i_{P,f}(X)$ to denote the i-fold application of $T_{P,f}$ starting on X. Further, due to Proposition 2 and the results of Tarski [25], $T_{P,f}$ has a least fixpoint which we denote by $T^\star_{P,f}$.

Proposition 3 *Let P be a simple program and f an y-rule interpretation on P. Then, $T^\star_{P,f}(\emptyset)$ is an f-rule model and for each $i \geq 0$, if X is an f-rule model then $T^i_{P,f}(\emptyset) \subseteq X$.*

Proof The first part, follows directly from the second equation of Lemma 2.

For the second part, the proof is by induction. For $i = 0$, the proposition vacuously holds. Suppose that the lemma holds for i and consider $I = T^i_{P,f}(\emptyset)$ and $J = T^{i+1}_{P,f}(\emptyset)$. By Proposition 2, $I \subseteq J$. Let X be a f-rule model. From the induction hypothesis, it follows that $I \subseteq X$. We consider two cases:

1. If $I = J$, then, trivially, $J \subseteq X$ and we are done.
2. Otherwise, $I \subset J$. Assume that, on the contrary $J \not\subseteq X$, i.e. $\exists a \cdot X(a) < J(a)$. From Lemma 2, we know that $\exists r \in P_a \cdot \mathcal{I}(I_{\models}(B(r)), X(a)) < f(r)$. From the induction hypothesis we have $I_{\models}(B(r)) \leq X_{\models}(B(r))$ and thus, because an implicator is decreasing in its first argument, we obtain that $\mathcal{I}(X_{\models}(B(r)), X(a)) \leq \mathcal{I}(I_{\models}(B(r)), X(a)) < f(r)$, and thus X is not an f-rule model, a contradiction.

□

Next, we define a variant of unfounded set w.r.t. a rule interpretation.

Definition 7 Let f be a rule interpretation of a simple program P and let I be a fuzzy interpretation. A set of literals X is an *f-unfounded set* w.r.t. I iff for each literal $l \in X$ and each rule $r \in P_l$:

$$B(r) \cap X \neq \emptyset \text{ or} \tag{2}$$

$$I(l) > I_s(f, r) \text{ or} \tag{3}$$

$$I_{\models}(B(r)) = 0 \tag{4}$$

holds. I is *f-unfounded-free* iff $\text{supp}(I) \cap X = \emptyset$ for each f-unfounded set X.

Clearly, if I is f-unfounded free and $\forall r \in P \cdot I_{\models}(r) \geq f(r)$, then I is unfounded free.

Proposition 4 *Let f be an y-rule interpretation. Then for each $i \geq 0$, $T^i_{P,f}(\emptyset)$ is f-unfounded-free.*

Proof We have two cases to consider, i.e. either $T^i_{P,f}(\emptyset) = \emptyset$ or $T^i_{P,f}(\emptyset) \neq \emptyset$. Clearly, the case $T^i_{P,f}(\emptyset) = \emptyset$ is trivial, as $\emptyset \cap X = \emptyset$, and thus $\emptyset$ is always f-unfounded-free.

For the case $T^i_{P,f}(\emptyset) \neq \emptyset$, let $J = T^i_{P,f}(\emptyset)$ and assume that J is not f-unfounded-free, i.e. there exists some unfounded set X w.r.t. J such that $\text{supp}(J) \cap X \neq \emptyset$.

Define the function $X_{\max}$ on interpretations by

$$X_{\max}(K) = \{K(a) \mid a \in X \wedge \forall b \in X \cdot K(b) \leq K(a)\} \ ,$$

i.e. $X_{\max}(K)$ contains the maximal value of K for the elements in X.

Let $m < i$ be the maximal number satisfying $X_{\max}(T^m_{P,f}(\emptyset)) < X_{\max}(J)$. Clearly, as $T_{P,f}$ is monotonic and we start our computation from $\emptyset$, such an m always exists. Using I to denote $T^m_{P,f}(\emptyset)$, this implies the existence of a literal $a \in X$ such that

$$I(a) < T_{P,f}(I)(a) = J(a) > 0 \ , \tag{5}$$

and, moreover,

$$\forall b \in X \cdot I(b) < T_{P,f}(I)(a) \ . \tag{6}$$

By Definition 6, we have

$$T_{P,f}(I)(a) = \sup_{r \in P_a} I_s(f, r) \ . \tag{7}$$

We consider the following partition, due to X being unfounded w.r.t. J, of the rules $r \in P_a$:

- If $B(r) \cap X \neq \emptyset$ then the implicator being residual, together with (6) and the properties of a t-norm, implies that $I_s(f, r) \leq I(B(r)) \leq I(X) < T_{P,f}(I)(a)$.
- If $J(B(r)) = 0$ then, because $T_{P,f}$ is monotonic, $I(B(r)) = 0$ and thus $I_s(f, r) = 0 < J(a) = T_{P,f}(I)(a)$.
- Otherwise, (5) and Definition 7 yield $T_{P,f}(I)(a) = J(a) > J_s(f, r)$. But $J_s(f, r) > I_s(f, r)$ because of Lemma 1 and $J \geq I$.

From the above, it follows that

$$\forall r \in P_a \cdot I_s(f, r) < T_{P,f}(I)(a) \ , \tag{8}$$

which, due to the fact that $\sup M \in M$ for any finite $M \subseteq \mathcal{L}$ in a total lattice $\mathcal{L}$, contradicts (7). □

Proposition 5 *Let I be a $\subset$-minimal fuzzy y-model for a simple program P. Then there exists a rule interpretation $f \in \mathcal{R}^y_{\mathcal{A}}$ such that $T^\star_{P,f}(\emptyset) = I$.*

Proof Since I is a fuzzy y-model, there exists a rule interpretation $f \in \mathcal{R}^y_{\mathcal{A}}$ such that $\forall r \in P \cdot I(r) \geq f(r)$. Proposition 3 then implies that $T^\star_{P,f}(\emptyset) \subseteq I$ and that $T^\star_{P,f}(\emptyset)$ is an f-rule model, and therefore a y-model. The minimality of I implies $T^\star_{P,f}(\emptyset) = I$. □

Theorem 1 *Any $\subset$-minimal x-consistent fuzzy y-model for P is an x-consistent fuzzy y-answer set for P.*

Proof Let I be a $\subset$-minimal x-consistent fuzzy y-model for P. By Proposition 5, $I = T^\star_{P,f}(\emptyset)$ for some rule interpretation f. On the other hand, Proposition 4 implies that $T^\star_{P,f}(\emptyset)$ is f-unfounded-free and thus, because $T^\star_{P,f}(\emptyset)$ is a $T_{P,f}$-fixpoint, unfounded-free. □

Note that for the total lattice $\mathcal{L} = [0, 1]$, the R-implicators defined in Section 2.1 always satisfy the residual principle.

Clearly, the reverse of Theorem 1 does not hold. However, if we restrict ourselves to $\subset$-minimal fuzzy y-answer sets, the reverse result trivially follows.

Theorem 2 *Any $\subset$-minimal x-consistent fuzzy y-answer set of P is a $\subset$-minimal x-consistent fuzzy y-model of P.*

Proof Let S be an $\subset$-minimal x-consistent fuzzy y-answer set of P and suppose that S is not a $\subset$-minimal x-consistent fuzzy y-model, i.e. there exists a fuzzy y-model $S' \subset S$. Take S', without loss of generality, such that it is $\subset$-minimal. Then, by Theorem 1, we have that S' is a fuzzy y-answer set, contradicting that S is $\subset$-minimal. □

6 Retrieving classical answer sets

The FASP framework presented in the previous section turns out to be a proper generalization of the classical answer set programming paradigm with the notions of fuzzy logic. First of all, ASP can be retrieved as a special case of FASP by choosing the truth lattice $\mathcal{L} = \{0, 1\}$.

Theorem 3 *Consider a program P and let $\mathcal{L}$ be the lattice $\{0, 1\}$. Furthermore let the aggregator $\mathcal{A}_c$ be such that $\mathcal{A}_c(\mathcal{B}_P, I_c) = 1$ iff $I_c(a) = 1$ for every atom $a \in \mathcal{B}_P$, and let the aggregator $\mathcal{A}_p$ be such that $\mathcal{A}_p(P, I_{\models}) = 1$ iff $I_{\models}(s) = 1$ for every rule $s \in P$. An interpretation M is an answer set of P* iff *the fuzzy interpretation f_M, with $f_M(l) = 1$* if *$l \in M$ and $f_M(l) = 0$* otherwise*, is a 1-consistent fuzzy 1-answer set of P.*

Proof Let M be a classical interpretation, and let f_M be its fuzzy variant on $\{0, 1\}$ defined by $f_M(l) = 1$ if $l \in M$ and $f_M(l) = 0$ otherwise.

Clearly, M is consistent iff f_M is 1-consistent and M is a model of P iff f_M is a fuzzy 1-model of P.

Hence we need to show that on the lattice $\{0, 1\}$, the classical definition of unfounded set coincides with the one of Definition 4. As the first condition is the same in both definitions, we only have to show that $M \not\models \beta$ is the same as condition two and three in Definition 4, i.e. either

$$f_M(a) > \inf\{y \mid \mathcal{I}(f_{M_{\models}}(\beta), y) \geq f_{M_{\models}}(a \leftarrow \beta)\}, \text{ or}$$

$$f_{M_{\models}}(\beta) = 0$$

holds on the lattice $\{0, 1\}$. From $M \not\models \beta$ we derive that there exists at least one $x \in \beta$ such that $M \not\models x$, which by construction of f_M yields that $f_M(x) = 0$, thus $f_{M_{\models}}(\beta) = 0$. For the reverse direction, we still have to show that the condition $f_M(a) > f_{M_s}(a \leftarrow \beta)$ results in $M \not\models \beta$. However, when this condition is satisfied on $\{0, 1\}$, we can only have one possibility, i.e. $f_M(a) = 1$ and $f_{M_s}(a \leftarrow \beta) = 0$. As $f_{M_{\models}}(a \leftarrow \beta) = 1$, it must be that $f_{M_{\models}}(\beta) = 0$; because $f_{M_{\models}}(\beta) = 1$ would yield $f_{M_s}(a \leftarrow \beta) = 1$. Now, $f_{M_{\models}}(\beta) = 0$ implies that there exists at least one $x \in \beta$ such that $f_M(x) = 0$, yielding that $M \not\models x$ by construction of M and f_M; and thus $M \not\models \beta$. □

Note that 1-consistency is needed to forbid (classical) contradictions, and the restriction to fuzzy 1-answer sets is mandated by the need to classically satisfy all rules and have the foundedness property of answer sets.

Example 18 Reconsider the program from Example 3. The empty set is not a 1-model of this program as it satisfies neither of the rules to degree 1. $K = \{(a, 1)\}$, $L = \{(b, 1)\}$, and $M = \{(a, 1), (b, 1)\}$ are 1-models. The latter is obviously not unfounded-free, while the first two are unfounded-free, i.e. K and L are both 1-consistent fuzzy 1-answer sets.

In the proposition above, no choice for $\mathcal{N}_c$, $\mathcal{N}_n$, $\mathcal{T}_c$, $\mathcal{T}_a$, and $\mathcal{I}$ is specified as all negators, t-norms and implicators on $\{0, 1\}$ coincide. However, when we allow for intermediate truth values, a choice for logical operators opens up. Below we argue that certain choices are more "answer set behaved" than others.

Classical answer sets cannot contain both a and $\neg a$. If one wants to preserve this behavior for fuzzy answer sets, i.e. such that a 1-consistent fuzzy answer set can not contain a and $\neg a$ simultaneously, not even to some degree, $\mathcal{T}_c$ should be chosen with care. E.g., on $\mathcal{L} = [0, 1]$, take $\mathcal{T}_c = \mathcal{T}_{\mathrm{W}}$ and consider the fuzzy interpretation $I = \{(a, 0.4), (\neg a, 0.4)\}$. Then, $\mathcal{T}_c(I(a), I(\neg a)) = \max(0.4 + 0.4 - 1, 0) = 0$. For this t-norm it holds, in general, that $\mathcal{T}_c(I(a), I(\neg a)) = 0$ iff $I(a) + I(\neg a) \leq 1$, which certainly does not correspond to a classical answer set semantics. However, there exist alternative choices for $\mathcal{T}_c$ that do not suffer from this problem, i.e. for which $\mathcal{T}_c(I(a), I(\neg a)) = 0$ iff $I(a) = 0$ or $I(\neg a) = 0$. Both $\mathcal{T}_M$ and $\mathcal{T}_P$ are such t-norms, and can be used to retrieve fuzzy answer sets with a classical ASP consistency notion.

Next, we consider the possible choices for the implicator. By definition, an implicator satisfies $\mathcal{I}(0, 0) = \mathcal{I}(0, 1) = \mathcal{I}(1, 1) = 1$ and $\mathcal{I}(1, 0) = 0$, which implies that any choice for $\mathcal{I}$ is sufficient to retrieve classical answer sets from the 1-consistent fuzzy 1-answer sets. However, when intermediate truth values are considered, certain choices for $\mathcal{I}$ are more answer set alike, as witnessed by the following example.

Example 19 Reconsider the program from Example 3 and consider the fuzzy interpretation $J = \{(a, 0.6), (b, 0.4)\}$. When using $\mathcal{I}_{S_M}$, we get $J_{\models}(r_1) = \max(1 - 0.6, 0.6) = 0.6$ and $J_{\models}(r_2) = \max(1 - 0.4, 0.4) = 0.6$, yielding that J will be at most a 0.6-answer set when inf is used as the aggregator. However, in a classical answer set context this looks a bit unintuitive as the heads of both rules are satisfied to exactly the same degrees as to which their bodies are applicable, and thus intuitively the rules should be totally satisfied. Applying $\mathcal{I}_{T_M}$ on the program yields $J_{\models}(r_1) = J_{\models}(r_2) = 1$, which fits that intuition.

The implicator $\mathcal{I}_{T_M}$ belongs to the class of R-implicators, for which, in general, it holds that $\mathcal{I}(x, y) = 1$ whenever $x \leq y$. All R-implicators in Example 2 satisfy the residuation principle or adjoint condition, i.e. $\mathcal{T}(x, y) \leq z$ iff $x \leq \mathcal{I}_{\mathcal{T}}(y, z)$ for all x, y, and z in $\mathcal{L}$. Note that for such implicators there is a direct expression to compute the support of a rule, namely $I_s(l \leftarrow \beta) = \mathcal{T}(I_{\models}(l \leftarrow \beta), I_{\models}(\beta))$.

Finally, to preserve classical answer set semantics, an interpretation should be said to satisfy a program to degree 1 iff it satisfies all rules of the program to degree 1. The aggregator $\mathcal{A}(P, I_{\models}) = \inf\{I_{\models}(s) \mid s \in P\}$ we introduced before satisfies this condition and can be used to retrieve classical answer sets.

7 Related work

Logic programming in the presence of uncertainty or imprecision has received a considerable amount of attention (see e.g. [1, 8] for overviews). It is however interesting to observe that the well known existing frameworks, including those that consider fuzzy interpretations, hold on to two valued concepts of rule satisfaction, model, etc. in the sense that a fuzzy interpretation satisfies a rule or not, it is a model or not, etc. This clearly sets them apart from the approach introduced in this paper.

The enrichment of ASP with concepts from fuzzy logic as well as from the closely related possibilistic logic [11] has been studied from various angles already. A particularly interesting approach is annotated answer set programming [24]. Here a rule is of the form

$$l\{f(z_1, z_2, \ldots, z_n)\} \leftarrow l_1\{z_1\}, l_2\{z_2\}, \ldots, l_n\{z_n\} \ ,$$

where $l, l_1, l_2, \ldots, l_n$ denote literals and $z_1, z_2, \ldots, z_n$ are annotation terms that can be understood as truth degrees. Such a rule asserts that l is true at least to degree $f(z_1, z_2, \ldots, z_n)$ whenever l_i is true at least to degree z_i (for $i = 1 \ldots n$). Because of this early revertment to the two-valued case, the approach in [24] does not consider to compute the actual degree to which a fuzzy interpretation is an answer set.

The implication based approach in [17] adheres closer to ours. A rule of the form $\alpha \overset{z}{\leftarrow} \beta$ is said to be satisfied by the interpretation iff (in our notation) $I_{\models}(\alpha) \geq_{\mathcal{L}} \mathcal{T}(I_{\models}(\beta), z)$. The residuation principle reveals a clear connection with our approach when committing to an R-implicator $\mathcal{I}_{\mathcal{T}}$: namely that I satisfies the rule $\alpha \overset{z}{\leftarrow} \beta$ according to [17] iff I satisfies this rule at least to degree z in our approach. This is also in accordance with [9] where the use of adjoint pairs $(\mathcal{T}, \mathcal{I}_{\mathcal{T}})$ is strongly advocated to preserve important theoretical results. Being able to impose specific satisfaction requirements for individual rules is in general an interesting feature, e.g., when rules and facts originate from different knowledge bases that are not all equally trusted. Note that this can be easily incorporated in our approach by choosing a suitable aggregator $\mathcal{A}$, which gives more weight to rules coming from sources that are more trusted.

In a similar way, [29] can be seen as a special case of the FASP framework presented in this paper as [29] commits itself, with limited motivation, to very specific choices for the user-selectable operators on the lattice [0, 1]. Some of these choices are at least questionable. E.g., using the Gödel negator $\mathcal{N}_g$ for interpreting naf yields that a rule $a \leftarrow$ not b will not be applied in any way although b is only true to a small degree, e.g. 0.1. Using the standard negator $\mathcal{N}_s$, as we do in our examples, this would yield a rule that is applicable to a degree 0.9, and, if a rule satisfaction of at least 0.8 is wanted with e.g. $\mathcal{I}_{S_M}$, we have $\max(0.1, y) = 0.8$, which implies that a will be derived at degree 0.8 in a fuzzy answer set.

A possibilistic definite logic program [18, 19] consists of rules annotated with certainty degrees. These degrees are used to establish a possibility distribution on the universe of atom sets, from which a possibilistic model is derived. The authors choose implicitly for the Gödel negator, as they first compute the classical answer sets, and afterwards compute, for an answer set S, the possibility to which each literal l is contained in S.

8 Conclusions and future research

There are many ways to increase the expressive power of answer set programming (ASP) by enriching it with mechanisms to deal with imprecision and uncertainty. In this paper we presented a general and elegant fuzzification of ASP, called fuzzy answer set programming (FASP). The generality is reflected in a high configurability by the user, which allows the system to be tailored to the application at hand. Among other things, the ability to choose an aggregator allows for future extensions of the semantics, e.g. incorporating rule preferences on fuzzy programs. The elegance is due to a close adherence to both the fuzzy logic and the answer set programming paradigm: as opposed to other approaches, FASP does not revert soon to the two-valued case but instead allows to compute the actual degree to which a fuzzy interpretation is an answer set. Furthermore we have shown that FASP extends the traditional answer set semantics and we have discussed choices of the fuzzy logical operators that behave more answer set like than others.

Clearly, there are a lot of topics that still need to be investigated, e.g. a fixpoint characterization, the complexity of the semantics, the use of disjunction etc., all parameterized by the choice of the (lattice) operations. In addition, we intend to explore natural FASP applications areas such as "web of trust", diagnosis, and decision support.

Acknowledgements Davy Van Nieuwenborgh is supported by the Flemish Fund for Scientific Research (FWO-Vlaanderen).

References

1. Alsinet, T., Godo, L., Sandri, S.: Two formalisms of extended possibilistic logic programming with context-dependent fuzzy unification: a comparative description. Electr. Notes Theor. Comput. Sci. **66**(5), 1–21 (2002)
2. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: Proceedings of the International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series (2003)
3. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge Press (2003)
4. Birkhoff, G.: Lattice Theory, vol. 25, 3rd edn. American Mathematical Society Colloquium Publications (1967)
5. Brewka, G.: Logic programming with ordered disjunction. In: Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 2002, pp. 100–105. AAAI Press (2002)
6. Brewka, G., Eiter, T.: Preferred answer sets for extended logic programs. Artif. Intell. **109**(1–2), 297–356 (April 1999)
7. Buccafurri, F., Leone, N., Rullo, P.: Strong and weak constraints in disjunctive datalog. In: Proceedings of the 4th International Conference on Logic Programming (LPNMR '97), pp. 2–17 (1997)
8. Damasio, C.V., Pereira, L.M.: Sorted monotonic logic programs and their embedding. In: Proceedings of the 10th Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-04), pp. 807–814 (2004)
9. Damasio, C.V., Medina, J., Ojeda-Aciego, M.: Sorted multi-adjoint logic programs: termination results and applications. Lect. Notes Comput. Sci. **3229**, 252–265 (2007)
10. De Vos, M., Vermeir, D.: On the role of negation in choice logic programs. In: Logic Programming and Non-Monotonic Reasoning Conference (LPNMR'99). LNAI, vol. 1730, pp. 236–246. Springer (1999)

11. Dubois, D., Prade, H.: Possibilistic logic: a retrospective and prospective view. Fuzzy Sets Syst. **144**(1), 3–23 (2004)
12. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: The diagnosis frontend of the dlv system. AI Commun. **12**(1–2), 99–111 (1999)
13. Gabbay, D., Laenens, E., Vermeir, D.: Credulous vs. sceptical semantics for ordered logic programs. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pp. 208–217. Morgan Kaufmann (1991)
14. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Logic Programming, Proceedings of the Fifth International Conference and Symposium, August 1988, pp. 1070–1080. Seattle, Washington, The MIT Press. (1988)
15. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**(3–4), 365–386 (1991)
16. Dix, D.N.J., Kuter, U.: Planning in answer set programming using ordered task decomposition. In: Proceedings of the 27th German Annual Conference on Artificial Intelligence (KI '03). LNAI, vol. 2821, pp. 490–504. Springer (2003)
17. Mateis, C.: Extending disjunctive logic programming by t-norms. In: Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR99), LNAI, vol. 1730, pp. 290–304. Springer (1999)
18. Nicolas, P., Garcia, L., Stéphan, I.: Possibilistic stable models. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 248–253 (2005)
19. Nicolas, P., Garcia, L., Stéphan, I., Lefèvre, C.: Possibilistic uncertainty handling for answer set programming. Ann. Math. Artif. Intell. **47**(1–2), 139–181 (2006)
20. Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M.: An a-prolog decision support system for the space shuttle. In: Third International Symposium on Practical Aspects of Declarative Languages. LNCS, vol. 1990, pp. 169–183. Springer (2001)
21. Novák, V., Perfilieva, I., Močkoř, J.: Mathematical Principles of Fuzzy Logic. Kluwer (1999)
22. Sacca, D., Zaniolo, C.: Stable models and non-determinism in logic programs with negation. In: PODS '90: Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 205–217. ACM Press (1990)
23. Soininen, T., Niemelä, I.: Developing a declarative rule language for applications in product configuration. In: Proceedings of the 1st International Workshop on Practical Aspects of Declarative Languages (PADL '99). LNCS, vol. 1551, pp. 305–319. Springer (1999)
24. Straccia, U.: Annotated answer set programming. In: Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-06) (2006)
25. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pac. J. Math. **5**, 285–309 (1955)
26. van Gelder, A., Ross, K.A., Schlipf, J.S.: Unfounded sets and well-founded semantics for general logic programs. In: Proceedings of the 7th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 1988, pp. 221–230. ACM Press, Austin, Texas, (1988)
27. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. Assoc. Comput. Mach. **38**(3), 620–650 (1991)
28. Van Nieuwenborgh, D., Vermeir, D.: Preferred answer sets for ordered logic programs. Theory Pract. Log. Program. **6**(1–2), 107–167 (2006)
29. Wagner, G.: A logical reconstruction of fuzzy inference in databases and logic programs. In: Proceedings of the International Fuzzy Set Association World Congress (IFSA'97) (1997)
30. Yager, R.: Including importances in owa aggregations using fuzzy systems modeling. IEEE Trans. Fuzzy Syst. **6**(2), 286–291 (1998)
31. Zadeh, L.: Fuzzy logic and approximate reasoning. Synthese **30**, 407–428 (1975)