Fuzzy answer set programming with literal preferences

Jeroen Janssen[1*] Steven Schockaert[2†] Dirk Vermeir[1] Martine De Cock[3‡]

1. Department of Computer Science, Vrije Universiteit Brussel, Belgium
2. Department of Applied Mathematics and Computer Science, Universiteit Gent, Belgium
3. Institute of Technology, University of Washington, Tacoma, USA
Email: {jeroen.janssen,dvermeir}@vub.ac.be, steven.schockaert@ugent.be, mdecock@u.washingon.edu

Abstract— In the current approaches to fuzzy answer set programming (FASP) one can state preferences amongst rules to denote their relative importance. However, in many situations we need more complex preferences such as those in the answer set optimization framework proposed by Brewka for crisp answer set programming. Unfortunately, these complex preferences do not readily fit into the current FASP approaches. In this paper we propose a language to state such preferences and show that programs with these preferences can be translated into equivalent general fuzzy answer set programming (gFASP) programs. This not only provides an implementation method, but also shows that this extension can be added as syntactic sugar on top of general fuzzy answer set programming solvers.

Keywords— Logic Programming, Fuzzy Logic, Answer Set Programming, Preference-based Logic Programming.

1 Introduction

Answer set programming [1] is a logic programming language based on the stable model semantics [2]. Roughly speaking, in answer set programming a program contains a set of *rules* of the form $a \leftarrow b_1, \ldots, b_n, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_m$ describing a problem whose solution is generated by an *answer set solver*. For example, if one is interested in finding solutions to the problem of assigning a seat s to either of two persons p_1 and p_2, the following program could be used:

$$\begin{array}{lcl} sit(p_1, s) & \leftarrow & \mathit{not}\ sit(p_2, s) \\ sit(p_2, s) & \leftarrow & \mathit{not}\ sit(p_1, s) \end{array}$$

This program has two *answer sets*, which correspond to the solutions of the modeled problem, $S_1 = \{sit(p_1, s)\}$ and $S_2 = \{sit(p_2, s)\}$.

In recent work, answer set programming and, more general, logic programming have been extended to deal with fuzzy [3, 4], probabilistic [5] and many-valued logics [6, 7, 8, 9], allowing for a more flexible description of both the problem domain and the proposed solutions of the problem domain. Specifically, in [4] degrees are attached to answer sets describing the quality of prospective solutions. For example, in case one is interested in only allowing solutions in which people are seated close to friends, one could write a program containing fuzzy *constraints*, as follows:

$$0 \leftarrow sit(p_1, s_1) \wedge sit(p_2, s_2) \wedge \mathit{friend}(p_1, p_2) \wedge \sim near(s_1, s_2)$$

where $sit(p, s)$ means that person p sits on seat s, $\mathit{friend}(p_1, p_2)$ that person p_1 is a friend of p_2 (to a certain degree), $near(s_1, s_2)$ denotes the proximity of seats s_1 and s_2 and $\sim$ is a negator. The constraint above would then attach a higher degree of quality to answer sets in which friends are close together. Note that $\sim near(s_1, s_2)$ should be used and not $near(s_1, s_2)$ since the satisfaction of the constraint rule should increase with the value of $near(s_1, s_2)$. More details can be found in Section 2.

Although the quality degree allows to express preference amongst solutions, the current proposals only allow this preference to be based on the degree of rule fulfillment. However, in many practical situations, we would like to express preferences based on the truth degrees of literals. Furthermore, we might want to express that the importance of some preferences depends on the context, i.e. we would like to have conditional preferences. For example, we wish to be able to state that if we are sitting on a seat, we prefer it to be by the window, but only consider this important when the scenery is nice, as follows:

$$scene : (sit(p, s) \rightarrow nearWin(s)) \triangleleft nicescenery \qquad (1)$$

where $nearWin(s)$ denotes that seat s is near the window, the $nicescenery$ literal determines the weight of importance of the rule and $\rightarrow$ is an implicator. The exact semantics of such rules are given in Section 3.

In the crisp case, Brewka proposed rules like (1) in [10]. However, to the best of our knowledge no such proposal exists for fuzzy answer set programs, which we remedy in this paper. The main contributions of this paper are the following:

- In Section 3 we propose a generalisation of the *Preference Description Language* (PDL) of Brewka to the fuzzy case.
- In Section 4 we show how programs containing these preferences can be translated to standard general fuzzy answer set programming (gFASP) programs [11], providing an implementation method and hence also showing that this extension is actually syntactic sugar on the general fuzzy answer set programming language.

2 General Fuzzy Answer Set Programming

General fuzzy answer set programming (gFASP) programs over a complete lattice $\mathcal{L}$ consist of *rules*, which are objects of the form

$$r\colon a \leftarrow f(b_1, \ldots, b_n)$$

*Funded by a joint Research Foundation–Flanders (FWO) project

†Postdoctoral fellow of the Research Foundation–Flanders (FWO)

‡On leave from Ghent University

ISBN: 978-989-95079-6-8

where r is the label of the rule, a and b_i, $1 \leq i \leq n$, are either elements from $\mathcal{L}$ or propositional symbols called *literals* (where $b_i = b_j$ for $i \neq j$ is possible), f is a (computable) function from $\mathcal{L}^n$ to $\mathcal{L}$ such that for $b_k, b'_k \in \mathcal{L}$ it holds that if $b_k \leq b'_k$, either $f(b_1, \ldots, b_k, \ldots, b_n) \leq f(b_1, \ldots, b'_k, \ldots, b_n)$ or $f(b_1, \ldots, b_k, \ldots, b_n) \geq f(b_1, \ldots, b'_k, \ldots, b_n)$ and $\leftarrow$ denotes a residual implicator over $\mathcal{L}$. If $a \in \mathcal{L}$, the rule is called a *constraint*. The head a of a rule r is denoted as r_h and the body $f(b_1, \ldots, b_n)$ is denoted as r_b.

The semantics are given by *interpretations*, i.e. fuzzy sets over the literals of a program. These are extended to give meaning to rules in a straightforward fashion, i.e. if I is an interpretation, $[a \leftarrow f(b_1, \ldots, b_n)]_I = [a]_I \leftarrow [f(b_1, \ldots, b_n)]_I$, where $[a]_I = I(a)$ if a is a literal and $[a]_I = a$ if $a \in \mathcal{L}$, and $[f(b_1, \ldots, b_n)]_I = f([b_1]_I, \ldots, [b_n]_I)$.

Models of a program P are defined using an *aggregator expression* $\mathcal{A}_P$ which maps *rule propositions* (corresponding to rule labels) to a value in a quasi-ordered set $\mathcal{Q}$. For example, given a program P_1 with two rules r_1 and r_2, a possible aggregator would be $\mathcal{A}_{P_1} = r_1 + r_2$. Such an expresssion can be evaluated by an interpretation, e.g. for I an interpretation of P we obtain $[\mathcal{A}_{P_1}]_I = \gamma([r_1]_I) + \gamma([r_2]_I)$, with γ an order-preserving $\mathcal{L} \rightarrow \mathcal{Q}$ mapping. A *k-model*, $k \in \mathcal{Q}$, is then an interpretation for which $[\mathcal{A}_P]_I \geq k$.

k-Answer sets of a program P are k-models that contain the maximal amount of knowledge inferrable from a program through forward chaining, without resorting to external hypotheses, i.e. they are models for which the truth value of every literal is supported by the application of a rule in the program.

As an example, consider program P_{seat}, describing seating arrangements, with the following rulebase $\mathcal{R}_{P_{seat}}$ over the lattice $([0,1], \leq)$:

$$
\begin{array}{rrll}
gen\colon & sit(P,S) & \leftarrow_g & 1 \\
c_1\colon & 0 & \leftarrow_g & sit(P,S) \wedge sit(P',S) \wedge P \neq P' \\
c_2\colon & 0 & \leftarrow_g & sit(P,S) \wedge sit(P,S') \wedge S \neq S' \\
cr\colon & 0 & \leftarrow_g & sit(P,S) \wedge (1 - sit(P,S)) \\
u\colon & unh(P) & \leftarrow_g & sit(P,S) \wedge friend(P,P') \\
 & & & \wedge sit(P',S') \wedge (1 - near(S,S')) \\
q\colon & 0 & \leftarrow_l & unh(P) \\
s\colon & s(P) & \leftarrow_g & sit(P,S) \\
a\colon & 0 & \leftarrow_g & 1 - s(P)
\end{array}
$$

where $\wedge$ is the minimum t-norm over $([0,1,\leq)$ and $\leftarrow_l$, $\leftarrow_g$ are resp. the Łukasiewicz and Gödel implicator. The aggregator is defined as $\mathcal{A}_{P_{seat}} = \inf_{r \in \mathcal{R}_{P_{seat}} \setminus \{gen\}} r$. Due to the fact that the gen-rule is not incorporated in the aggregation expression, this rule generates random truth degrees for $sit(P,S)$. To ensure that $sit(P,S)$ is either 0 or 1, we use the cr (crispify) rule. Rules c_1 and c_2 guarantee that a seat is occupied by a single person, resp. that a person only occupies one seat. The a and s rules ensure that everyone has a seat and the u and q rules are used to prefer solutions where friends are seated close together. Note that due to grounding a rule such as $s(P) \leftarrow_g sit(P,S)$ actually denotes a set of rules $\{s(p_i) \leftarrow_g sit(p_i, s_j)\}$ for a set of persons $(p_i)_{i \in I}$ and a set of seats $(s_j)_{j \in J}$.

If we have three seats s_1, s_2 and z, only two of which are near, viz. $near(s_1, s_2)^{\cdot 8}$ and $near(s_2, s_1)^{\cdot 8}$ and there are three persons a, b and c connected by friendship degrees $friend(a,b)^{\cdot 8}$, $friend(a,c)^{\cdot 5}$, and $friend(b,c)^0$, then from program P_{seat} one could e.g. derive the .2-answer set $A_1 = \{sit(a,s_1)^1,\ sit(b,z)^1,\ sit(c,s_2)^1,\ unh(a)^{\cdot 8},\ \ldots\}$ and the (better) .5-answer set $A_2 = \{sit(a,s_1)^1,\ sit(b,s_2)^1,\ sit(c,z)^1,\ unh(a)^{\cdot 5}, \ldots\}$.

For more details on the gFASP framework, we refer the reader to [11].

3 Complex Preferences on Literals

3.1 Fuzzy preference rules: syntax

In this section, we build a fuzzy preference framework based on a *fuzzy preference description language* ($FPDL$) which is an extension of the language PDL proposed by Brewka in [10]. Formally, a gFASP program with literal preferences consists of a tuple (P, e), where P is a gFASP program and e is an expression from $FPDL$. This expression determines a preference ordering on answer sets based on truth degrees of literals.

The basic elements of this language are *fuzzy preference rules*. Note that although they are called *rules*, they should not be confused with the rules of a fuzzy answer set program introduced in Section 2: the former are used for creating preference expressions (the e in (P, e)), whereas the latter are used to create (general) fuzzy answer set programs (the P in (P, e)). Formally the fuzzy preference rules are defined over a set of literals L and the complete lattice $\mathcal{L}$ as expressions of the following form:

$$r : g(a_1, \ldots, a_n) \triangleleft f(b_1, \ldots, b_m) \tag{2}$$

In this rule, the a_is ($1 \leq i \leq n$) and b_js ($1 \leq j \leq m$) are literals from L, the function g is a mapping from $\mathcal{L}^n$ to $\mathcal{L}$, f is a mapping from $\mathcal{L}^m$ to $\mathcal{L}$ and r is the label of the rule. For convenience, for a given fuzzy preference rule r we also refer to $g(a_1, \ldots, a_n)$ as r_h (the *head* of the rule) and to $f(b_1, \ldots, b_m)$ as r_b (the *body* of the rule).

Intuitively, the g-function denotes an expression determining the *suitability* of an answer set with respect to a certain preference and the f-function denotes the *applicability* of the rule, meaning the conditions under which this preference rule is considered important or relevant. For example, consider the following preference rule:

$$scene : (sit(p,s) \rightarrow nearWin(s)) \triangleleft nicescenery$$

This rule states that by default we (with "we" being person p) would like to have a seat near the window. However, we only consider this requirement important to the degree that the scenery is actually nice. The reason for choosing g to be an implication in this rule is that we want to state that this specific rule is also fulfilled (thus the answer set considered is still "suitable") if we are not sitting at this seat. Note that there is a difference between the $\rightarrow$ in the head and the $\triangleleft$ of the rule: if we would rewrite the scene rule as

$$scene' : (sit(p,s) \wedge nicescenery \rightarrow nearWin(s)) \triangleleft 1$$

it would denote that the nicer the scenery, the closer to the window we would like to sit rather than that we only find it important to sit near the window when the scenery is nice.

ISBN: 978-989-95079-6-8

This can also be seen from the fact that if $\rightarrow$ is a residual implicator and $nicescenery$ is true to a degree of 0.5, the $scene$ rule is only fully fulfilled when $nearWin(s)$ is 1 (or $sit(p,s)$ is 0), whereas in the $scene'$ rule it suffices that $nearWin(s)$ is 0.5 (or $sit(p,s)$ is 0).

Intuitively we thus prefer answer sets with a better suitability score, but can forgive the answer set a low suitability score if the applicability degree is rather low. Formally the semantics of rules are dependent on a function $suit$ and app defined for a given fuzzy preference rule r and an arbitrary answer set A as

$$suit(A,r) = [g(a_1,\ldots,a_n)]_A = g(A(a_1),\ldots,A(a_n))$$

$$app(A,r) = [f(b_1,\ldots,b_m)]_A = f(A(b_1),\ldots,A(b_m))$$

In the approach of Brewka [10], the semantics of preference rules are defined by attaching a *penalty* score in $\mathbb{R}$ to each applicable rule. Such a penalty reflects how good an answer set fulfills a preference rule, where lower penalties correspond to better fulfillment. Inapplicable rules are not considered important and are therefore attached penalty 0, i.e. they are always optimally fulfilled.

Unfortunately, the same approach is not possible when generalizing from crisp to fuzzy answer set programming: in the fuzzy case the applicability value is in $[0,1]$ and by reducing the values $suit(A,r)$ and $app(A,r)$ to a single number (i.e. penalty score in $\mathbb{R}$), we lose the important semantic distinction between suitability of answer sets and applicability of preference rules. In the following sections, we define an alternative semantics for fuzzy preference rules which does adhere to this distinction. We will first give the semantics for rules with a crisp applicability score and then define penalty expressions built from rules; starting from the semantics thus obtained we then introduce the semantics for rules with a fuzzy applicability score using the recently introduced concept of gradual number [12]; finally we introduce the general penalty expressions built from rules with fuzzy applicabilities.

3.2 Rules with crisp applicability

Suppose r is a rule with a crisp applicability condition (i.e. $app(A,r)$ is either 1 or 0), then we define the penalty score as:

$$pen^c(A,r) = \begin{cases} \Xi(suit(A,r)) & \text{if } app(A,r) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where Ξ is a decreasing $\mathcal{L}$ to $[0,+\infty[$ mapping, such that $\Xi(1) = 0$, in the examples in this paper usually assumed to be $\Xi(x) = 1 - x$.

As an example, consider the following rule:

$$VIP_p : goodSeat(p) \triangleleft vip(p)$$

where $goodSeat(p) \in [0,1]$ reflects how good the seat of person p is and $vip(p) \in \{0,1\}$ denotes whether person p is a VIP. This rule encodes that we find it important that VIPs have good seats. If $A(goodSeat(p)) = 0.8$ and $A(vip(p)) = 1$, we know from the definition of pen^c that $pen^c(A,r) = 1 - 0.8 = 0.2$. Hence, answer set A has a low penalty score and thus represents a seating configuration that satisfies our wishes.

Starting from rules with crisp applicability, we can define the set of complex penalty expresssions $FPDL^p$ as follows:

1. If r is a fuzzy preference rule whose applicability condition takes only crisp values, r is in $FPDL^p$;
2. If $e_1, e_2, \ldots, e_k$ are in $FPDL^p$, then so are $(sum\ e_1,\ldots,e_k)$ and $(max\ e_1,\ldots,e_k)$.

We extend the definition of pen^c to cover complex penalty expressions in a straightforward way, e.g.

$$pen^c(A, sum\ e_1, ..., e_k) = pen^c(A,e_1) + ... + pen^c(A,e_k)$$

For example, suppose we have two VIP_p rules for persons p_1 and p_2, then we can combine these into one penalty expression summing the penalties as

$$e_v : (sum\ VIP_{p_1}, VIP_{p_2})$$

If we then have an answer set A for which $pen^c(A, VIP_{p_1}) = 0.2$ and $pen^c(A, VIP_{p_2}) = 0.5$, the total penalty of this expression is $pen^c(A, (sum\ VIP_{p_1}, VIP_{p_2}) = 0.7$.

3.3 Rules with vague applicability

In the general case, where applicability conditions are vague, we define penalties as gradual numbers in the sense of [12], i.e. as mappings from $]0,1]$ to $\mathbb{R}$. For each threshold λ in $]0,1]$, we convert preference rules r of the form (2) to preference rules r_λ with a crisp applicability condition:

$$r_\lambda : g(a_1,\ldots,a_n) \triangleleft (f(b_1,\ldots,b_m) \geq \lambda) \tag{3}$$

The (gradual) penalty of A w.r.t. a preference rule r is then given by the gradual number $pen(A,r)$, defined for each λ in $]0,1]$ as

$$pen(A,r)(\lambda) = pen^c(A, r_\lambda) \tag{4}$$

In Figure 1, one can see an example of the penalty thus induced by an answer set A on the rule VIP_p where $A(vip(p)) = app(A, VIP_p) = 1/2$ and $A(goodSeat(p)) = suit(A, VIP_p) = 0.75$. Note that for each rule the set $\{\lambda \mid pen(A,r)(\lambda) = \Xi(suit(A,r))\}$ is the half-open interval $]0, app(A,r)]$ and the set $\{\lambda \mid pen(A,r)(\lambda) = 0\}$ is the half-open interval $]app(A,r), 1]$. Hence, an answer set is only accountable (gets a penalty) w.r.t. a rule to the extent that the rule is applicable for the answer set.

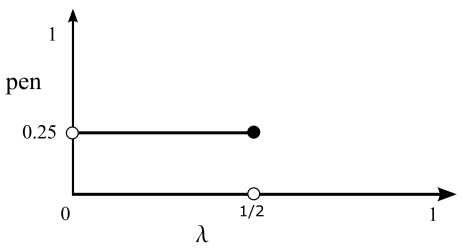

Figure 1: Example rule penalty

These penalties can easily be extended to complex preference expressions from a set $FPDL^{gp}$, defined analogously as $FPDL^p$ above, but starting from arbitrary preference rules. To define the semantics for expressions from $FPDL^{gp}$, we rely on the semantics of corresponding expressions from

ISBN: 978-989-95079-6-8

$FPDL^p$. Specifically, for e in $FPDL^{gp}$ and λ in $\mathcal{L}$, we define

$$pen(A, e)(\lambda) = pen^c(A, e^\lambda)$$

where e^λ is obtained from e by replacing all occurrences of preference rules r by the corresponding preference rules r_λ. Note that in principle, aggregation strategies that operate directly on gradual penalties could be defined as well.

Figure 2 depicts $pen(A, e)$ (the solid line) and $pen(A', e)$ (the dashed line) for an expression of the form $e = (sum\ dance, win)$ where the rule

$$dance : (sit(p, s) \rightarrow nearDanceFl(s)) \triangleleft nearFriends(p)$$

encodes that person p would like to sit near the dancefloor, but only finds this important if he is sitting close to his friends. The other rule is defined as

$$win : (sit(p, s) \rightarrow nearWin(s)) \triangleleft nearColleagues(p)$$

and denotes that person p likes a window seat, but only finds this important if he is sitting close to his colleagues. The answer sets A and A' take on the following applicability and suitability values:

app	$dance$	win
A	0.375	0.625
A'	0.75	0.5

$suit$	$dance$	win
A	0.18	0.82
A'	0.8	0.95

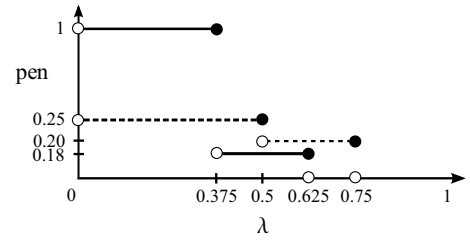

Figure 2: Example complex penalties

Now, given Figure 2, which answer set is preferred: A or A'? This depends on the application: if one only wants to look at the higher (more important) lambdas, we would prefer A as for $\lambda = 0.75$ this answer set has penalty 0; however if the global penalty score over all lambdas is important, we would prefer A' as A has a high penalty for values lower than 0.375 (viz. 1). Hence many options are viable, depending on the application. Furthermore there is a need to state preferences amongst expressions themselves so that we can e.g. state that only satisfying the $dance$ rule is better than only satisfying the win rule. In the next section we show how such orderings of answer sets based on the penalties and the preferences amongst preference expressions can be stated.

3.4 Preference strategies

Using penalties, we can define preference orderings on answer sets, which are referred to as *strategies*. Various such strategies can be specified. A natural strategy if we are comparing answer sets on the basis of one expression is to define the preference ordering $\leq$ between answer sets, based on an ordering $\preceq$ of gradual numbers. This strategy is denoted by $(\preceq\ e)$, and the corresponding preference ordering $\leq$ is defined for answer sets A_1 and A_2 as

$$A_1 \leq A_2 \equiv pen(A_1, e) \succeq pen(A_2, e)$$

Several partial and total orderings between gradual numbers can be used to this end. When the applicability values are interpreted as priorities, the total ordering $\preceq_1$ can be used, where $g_1 \preceq_1 g_2$ for gradual numbers g_1 and g_2 whenever

1. $g_1 = g_2$; or
2. $g_1(\lambda) < g_2(\lambda)$ for some λ in $]0, 1]$, and $g_1(\lambda') = g_2(\lambda')$ for all $\lambda' > \lambda$.

A useful partial ordering is the pointwise extension of the natural ordering $\leq$ on $\mathbb{R}$, denoted by $\preceq_2$, i.e. $g_1 \preceq_2 g_2$ iff $g_1(\lambda) \leq g_2(\lambda)$ for all λ in $]0, 1]$. Finally, an averaging strategy $\preceq_3$ could be used where $g_1 \preceq_3 g_2$ iff $\int_0^1 g_1(\lambda)w(\lambda)\,\mathrm{d}\lambda \leq \int_0^1 g_2(\lambda)w(\lambda)\,\mathrm{d}\lambda$. Intuitively, we may want to give more weight to the values of g_1 and g_2 for larger values of λ. This is encoded by using an increasing $]0, 1]$–$]0, 1]$ mapping w, defined, for instance, by $w(\lambda) = \lambda$ for all λ in $]0, 1]$.

As an example, consider Figure 2 with expression e from Section 3.3 again. From the picture one can see that if $\preceq_1$ is used, answer set A would be preferred over A' as for $\lambda = 0.75$ we have that $pen(A, e)(\lambda) < pen(A', e)(\lambda)$ and for any $\lambda > 0.75$ it holds that $pen(A, e)(\lambda) = 0 = pen(A', e)(\lambda)$. Hence $\preceq_1$ encodes a form of priority where we discriminate answer sets on their penalties for rules with a high applicability. Using $\preceq_2$, we would obtain that A and A' are incomparable. Thus $\preceq_2$ takes a more global approach, only considering preference when one of the two answer sets globally imposes a lower penalty score. With $\preceq_3$ and $w(\lambda) = \lambda$ for each $\lambda \in]0, 1]$ then, we obtain that A' would be preferred over A, due to the fact that A has a very high penalty at lower λ scores and is close to the penalty of A at the more important λ values. We can thus conclude that $\preceq_3$ can be used when one is not only interested in the fact that one penalty is higher than the other, but also in the exact value of these penalties.

The $\preceq_i$ $(1 \leq i \leq 3)$–strategies are not sufficient however: if we have two preference rules r_1 and r_2 and we want r_1 to be more important than r_2, we cannot readily encode this using a combination of the sum or max expressions and one of the orderings on the gradual penalties. To this end, we introduce other strategies, encoded as expressions from the set $FPDL$. First we have $(\preceq\ e) \in FPDL$ for $e \in FDPL^{gp}$ and $\preceq$ a partial ordering between gradual numbers. In analogy with Brewka [10], we also consider the following expressions:

1. If $e_1, \ldots, e_k$ are in $FPDL$, then $(pareto\ e_1, \ldots, e_k)$ and $(lex\ e_1, \ldots, e_k)$ are in $FPDL$
2. If $e_1, \ldots, e_k$ are in $FPDL^{gp}$ then $(rinc\ e_1, \ldots, e_k)$ and $(rcard\ e_1, \ldots, e_n)$ are in $FPDL$

The semantics of $(pareto\ e_1, \ldots, e_k)$ and $(lex\ e_1, \ldots, e_k)$ are defined analogously to the crisp case. In particular, the ordering $\leq$ induced by $(pareto\ e_1, \ldots, e_k)$ is defined by

$$A_1 \leq A_2 \equiv A_1 \leq_1 A_2 \wedge \ldots \wedge A_1 \leq_k A_2$$

ISBN: 978-989-95079-6-8

where $\preceq_i$ is the ordering induced by e_i. As an example, consider the expression $(pareto\ \ (\preceq_1\ \ dance), (\preceq_1\ \ win))$ with $dance$ and win as defined in Section 3.3. The preference ordering induced by this expression does not favor either of the expressions $(\preceq_1\ \ dance)$ and $(\preceq_1\ \ win)$.

Similarly, the ordering induced by $(lex\ e_1, \ldots, e_k)$ is given by

$$A_1 \leq A_2 \equiv \forall j \in 1..k \cdot A_1 \preceq_j A_2 \\ \vee \exists j \in 1..k \cdot A_1 \prec_j A_2 \wedge \forall j' < j \cdot A_1 \preceq_{j'} A_2$$

Expression $(lex\ (\preceq_1\ \ dance), (\preceq_1\ \ win))$ states that we first need to look whether we can prefer one of two answer sets A and A' by the expression $(\preceq_1\ \ dance)$. If this is not the case, we look whether $(\preceq_1\ \ win)$ can be used to prefer one of the two. Intuitively we thus prefer answer sets where person p is close to the dancefloor, even if that means he is sitting far from the window and close to his colleagues.

The $pareto$ and lex strategies both take arbitrary $FPDL$ expressions as their arguments. The next two strategies we introduce, $rinc$ and $rcard$, however, have expressions from $FPDL^{gp}$ as arguments. The difference with the $\preceq_i$ ($1 \leq i \leq 3$) strategies introduced earlier is that $rinc$ and $rcard$ base the preferences on more than one penalty expression and thus allow to have some interplay between rules and penalty expressions.

The semantics of $(rinc\ e_1, \ldots, e_k)$ and $(rcard\ e_1, \ldots, e_k)$ are defined in terms of the interval-valued fuzzy set P^a_A in the finite universe $\{1, \ldots, k\}$, where A is an answer set, $a \in [0, +\infty[$ and $L^a_A(i) = \{\lambda \mid pen(A, e_i)(\lambda) = a\}$:

$$P^a_A(i) = \begin{cases} [\inf L^a_A(i), \sup L^a_A(i)] & \text{if } L^a_A(i) \neq \emptyset \\ [0, 0] & \text{otherwise} \end{cases}$$

Intuitively, P^a_A is the set of (indices of) subexpressions e_i that attach a penalty a to an answer set A. Due to penalties being gradual numbers and because of the specific form of $pen(A, e)(\lambda)$ for any penalty expression e, P^a_A is an interval-valued fuzzy set.

The preference ordering induced by $(rinc\ \ e_1, \ldots, e_k)$ is defined as $A_1 \leq A_2$ iff

1. $P^a_{A_1} = P^a_{A_2}$ for all a in $[0, +\infty[$; or
2. $P^a_{A_1} \subset P^a_{A_2}$ for some a in $[0, +\infty[$, and $P^b_{A_1} = P^b_{A_2}$ for all $b < a$

where $\subset$ is an inclusion measure on interval-valued fuzzy sets, such as those proposed in [13]. As an example, consider $(rinc\ \ dance, win, boss)$ with $dance$ and win as in Section 3.4 and $boss$ the rule

$$boss : (sit(p, s) \rightarrow nearSpeechStand(s)) \triangleleft nearBoss(p)$$

Furthermore, we have answer sets A and A' with the following applicability and suitability values

app	r_1	r_2	r_3
A	0.25	0.5	0.5
A'	0.75	0.25	1

$suit$	r_1	r_2	r_3
A	0.5	0.5	0.25
A'	1	0.1	0.5

By definition of $rinc$ we first need to check whether we can discriminate between A and A' on the basis of P^0_A and $P^0_{A'}$. From the definition we easily obtain that:

P^0	r_1	r_2	r_3
A	$[0.25, 1]$	$[0.5, 1]$	$[0.5, 1]$
A'	$[0, 1]$	$[0.25, 1]$	$[0, 0]$

Note that $P^0_{A'}(1) = [0, 1]$ as the penalty of A' is 0 over the whole interval. Likewise $P^0_{A'}(3) = [0, 0]$ as the rule is fully applicable and penalty 0 is thus never reached. We use the inclusion measure $A \subseteq B \equiv \forall x \in X \cdot A(x) \leq B(x)$ for interval-valued fuzzy sets A and B over a universe X where $[a, b] \leq [a', b'] \equiv b \leq b' \wedge (b - a) \leq (b' - a')$ in this example, i.e. we take both the more important high λs and the amount of lambdas into account. By this definition $P^0_A(2) < P^0_{A'}(2)$ and $P^0_{A'}(3) < P^0_A(3)$ meaning that using $rinc$ the answer sets A and A' are incomparable.

For $(rcard\ e_1, \ldots, e_k)$, which uses a cardinality based approach, the ordering is given by $A_1 \leq A_2$ iff

1. $|P^a_{A_1}| = |P^a_{A_2}|$ for all a in $[0, +\infty[$; or
2. $|P^a_{A_1}| < |P^a_{A_2}|$ for some a in $[0, +\infty[$, and $|P^b_{A_1}| = |P^b_{A_2}|$ for all $b < a$

where $|U|$ denotes a measure of cardinality for the interval-valued fuzzy set U, such as those defined in [14]. Note that $rcard$ induces a partial order, whereas $rinc$ induces a total order.

As an example, consider $(rcard\ dance, win, boss)$, A and A' defined as before and a cardinality measure of the form $|A| = \sum_{x \in X} f(A(x))$ on an interval-valued fuzzy set A. When interval-valued membership degrees are used to encode uncertainty or bipolarity, $f_1([a, b]) = \frac{a+b}{2}$ is a natural choice. However, in our case, we need a notion of cardinality which is increasing in both b and $b - a$: a given penalty value is more representative for a penalty expression when it corresponds to more λ values, and to higher λ values. A suitable function is therefore given by $f_2([a, b]) = b(b - a)$. To further justify this choice, we provide the outcomes of both f_1 and f_2 in this example. For $rcard$ we obtain with f_1 that $|P^0_A| = \frac{4.25}{2} > \frac{2.25}{2} = |P^0_{A'}|$ whereas with f_2 we get $|P^0_A| = 1.75 = |P^0_{A'}|$, hence with f_1 answer set A would be preferred over A', whereas with f_2 no difference can be made by looking at penalty 0 and thus we must look for a higher penalty that does make a difference. The first penalty we should consider is 0.5 as this is the first penalty actually occurring for some rules. Computing $P^{0.5}$ gives:

$P^{0.5}$	r_1	r_2	r_3
A	$[0, 0.25]$	$[0, 0.5]$	$[0, 0]$
A'	$[0, 0]$	$[0, 0]$	$[0, 1]$

From this table we see that with f_2 we get $|P^0_A| = 0.25 * 0.25 + 0.5 * 0.5 = 0.3125$ and $|P^0_{A'}| = 1$, hence $A \leq A'$ and thus A' is preferred over A.

4 Translating Literal Preferences to a gFASP Program

In this section, we show that the gFASP framework we summarized in Section 2 is sufficiently general to encode $FPDL$–

ISBN: 978-989-95079-6-8

rules, an observation which immediately leads to an implementation method. Consider the gFASP program P, comprised of a rule base $\mathcal{R}_P$ and an aggregator expression $\mathcal{A}_P$ over the quasi-order $\mathcal{Q}_P$, and the expression e from $FPDL$. Specifically, we will show that a set of rules $\mathcal{R}_{P'}$ can be found, and an aggregator expression $\mathcal{A}_{P'}$ over a quasi-order $\mathcal{Q}_{P'}$, such that for any two answer sets A_1 and A_2

$$[\mathcal{A}_{P'}]_{A_1} \leq [\mathcal{A}_{P'}]_{A_2} \equiv [\mathcal{A}_P]_{A_1} \leq [\mathcal{A}_P]_{A_2} \wedge A_1 \leq_e A_2$$

Intuitively, $\mathcal{A}_P$ encodes which rules are more important, and how answer sets should be penalized when some rules are violated, while e encodes preferences between the actual solutions, i.e. preferences on the truth values of the literals.

As a first step, we need to provide the means to refer to the value of a certain function over literals $f(a_1, \ldots, a_n)$ from the aggregator expression. This is done by creating a constraint of the form $c_f : 0 \leftarrow \sim f(a_1, \ldots, a_n)$ where $\sim$ is an involutive negator and $\leftarrow$ is an implicator satisfying $0 \leftarrow x = \sim x$. It is easy to see that, in this way, the interpretation of rule proposition c_f, which we can refer to in the aggregator expression, will always be equal to the corresponding interpretation of $f(a_1, \ldots, a_n)$. Since the aggregator expression is increasing in all rule propositions, this procedure only works if we prefer answer sets with higher values of $f(a_1, \ldots, a_n)$. If, on the other hand, lower values of $f(a_1, \ldots, a_n)$ are preferred, we introduce the constraint $c_f : 0 \leftarrow f(a_1, \ldots, a_n)$ and use $\sim c_f$ in the aggregator.

Let $\mathcal{R}_e = \{r_1, \ldots, r_n\}$ be the set of preference rules occurring in e. Given that the preference of answer sets increases with increasing suitability of the rules and decreasing applicability, the aforementioned procedure leads to the rulebase $\mathcal{R}_{P'}$, defined by

$$\mathcal{R}_{P'} = \mathcal{R}_P \cup \{g_i : 0 \leftarrow \sim r_{ih} \mid i \in 1..n\} \cup \{f_i : 0 \leftarrow r_{ib}\}$$

The second part of our embedding consists of constructing a suitable aggregator expression $\mathcal{A}_{P'}$ and corresponding quasi-order $\mathcal{Q}_{P'}$. The aggregator $\mathcal{A}_{P'}$ is straightforwardly obtained from $\mathcal{A}_P$ by extending it with the various suitability and applicability values, i.e.:

$$\mathcal{A}_{P'} = (\mathcal{A}_P, (g_1, \ldots, g_n, \sim f_1, \ldots, \sim f_n))$$

The corresponding quasi-order is given by $\mathcal{Q}_{P'} = (\mathcal{Q}_P \times \mathcal{L}^{2n}, \leq_{\mathcal{Q}_{P'}})$ where

$$(a, b) \leq_{\mathcal{Q}_{P'}} (a', b') \equiv (a \leq_{\mathcal{Q}_P} a') \wedge (b \leq_e b')$$

Note how we can discriminate between answer sets on the basis of their rule fulfillment, using the first element of tuples from $\mathcal{Q}_{P'}$, as well as on the basis of the fulfillment of preference expression e, using the second element. The ordering $\leq_e$ is defined in the same way as the orderings on preference expressions, taking as arguments tuples with applicability and suitability scores, instead of answer sets.

The following proposition shows that the proposed embedding indeed preserves the intended semantics:

Proposition 1 *Let P be a gFASP program and e a fuzzy preference expression from FPDL. For the program P', obtained in the way described above, it holds for any two answer sets A_1 and A_2 that*

$$[\mathcal{A}_{P'}]_{A_1} \leq [\mathcal{A}_{P'}]_{A_2} \equiv [\mathcal{A}_P]_{A_1} \leq [\mathcal{A}_P]_{A_2} \wedge A_1 \leq_e A_2$$

5 Concluding remarks

In this paper, we generalized Brewka's answer set optimization framework to the fuzzy domain. The resulting formalism combines the flexibility of fuzzy answer set programming with the expressiveness of conditional preferences. We emphasized the conceptual difference between applicability and suitability of preference rules. This difference, which collapses in the two-valued case, played a crucial role in our fuzzification, leading to a generalization of penalty values using the notion of gradual numbers.

Although our extension is independent of any particular approach to fuzzy answer set programming, we showed that the recently introduced gFASP framework can provide a convenient implementation. One of the core elements in this approach is the use of an aggregator expression to discriminate between answer sets. We have shown how fuzzy preference expressions can be seen as particular choices of this aggregator expression. Hence, fuzzy preference expressions should be regarded as a convenient way to encode complex quasi-orders between answer sets, in a truely declarative way, rather than as a fundamental extension to this earlier work.

References

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP1988*, pages 1081–1086. MIT Press, 1988.

[3] T. Lukasiewicz and U. Straccia. Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web. In *Proc. of RR2007*, pages 289–298. Springer-Verlag, 2007.

[4] D. Van Nieuwenborgh, M. De Cock, and D. Vermeir. An introduction to fuzzy answer set programming. *Annals of Mathematics and Artificial Intelligence*, 50(3-4):363–388, 2007.

[5] T. Lukasiewicz. Probabilistic logic programming. In *Proc. of ECAI1998*, pages 388–392. J. Wiley and Sons, 1998.

[6] U. Straccia. Annotated answer set programming. In *Proc. of IPMU2006*, 2006.

[7] C.V. Damásio, J. Medina, and M. Ojeda-Aciego. Sorted multi-adjoint logic programs: termination results and applications. In *Proc. of JELIA2004*, pages 260–273. Springer-Verlag, 2004.

[8] C. V. Damásio and L. M. Pereira. Antitonic logic programs. In *Proc. of LPNMR2001*, pages 379–392. Springer-Verlag, 2001.

[9] U. Straccia. Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In *Reasoning Web, 4th International Summer School, Tutorial Lectures*, number 5224, pages 54–103. Springer Verlag, 2008.

[10] G. Brewka. Complex preferences for answer set optimization. In *Proc. of KR2004*, pages 213–223, 2004.

[11] J. Janssen, S. Schockaert, D. Vermeir, and M. De Cock. General fuzzy answer set programs. Accepted for WILF2009.

[12] J. Fortin, D. Dubois, and H. Fargier. Gradual numbers and their application to fuzzy interval analysis. *Fuzzy Systems, IEEE Transactions on*, 16(2):388–402, 2008.

[13] C. Cornelis and E.E. Kerre. Inclusion measures in intuitionistic fuzzy set theory. In *Proc. of ECSQARU2003*, pages 345–356, 2003.

[14] G. Deschrijver and P. Král'. On the cardinalities of interval-valued fuzzy sets. *Fuzzy Sets and Systems*, 158(15):1728–1750, 2007.

ISBN: 978-989-95079-6-8