Towards Possibilistic Fuzzy Answer Set Programming

Kim Bauters[*] and **Steven Schockaert**[†]
Universiteit Gent
Department of Applied Mathematics and Computer Science
Krijgslaan 281
9000 Gent, Belgium

Jeroen Janssen[‡] and **Dirk Vermeir**
Vrije Universiteit Brussel
Department of Computer Science
Pleinlaan 2
1050 Brussel, Belgium

Martine De Cock[§]
University of Washington
Institute of Technology
1900 Commerce Street
WA-98402 Tacoma, USA

Abstract

Fuzzy answer set programming (FASP) is a generalization of answer set programming to continuous domains. As it can not readily take uncertainty into account, however, FASP is not suitable as a basis for approximate reasoning and cannot easily be used to derive conclusions from imprecise information. To cope with this, we propose an extension of FASP based on possibility theory. The resulting framework allows us to reason about uncertain information in continuous domains, and thus also about information that is imprecise or vague. We propose a syntactic procedure, based on an immediate consequence operator, and provide a characterization in terms of minimal models, which allows us to straightforwardly implement our framework using existing FASP solvers.

Introduction

Answer set programming (ASP) is a form of non-monotonic reasoning which is based on the stable-model semantics (Gelfond and Lifzchitz 1988). Among others, ASP has proven successful as an elegant and convenient vehicle for commonsense reasoning in discrete domains, and to encode combinatorial optimization problems in a purely declarative way. Intuitively, an answer set program can be seen as a collection of rules on which forward chaining is applied. Consider, for instance, the following program:

$$rushHour \leftarrow \quad (1)$$

$$longExpectedDrivingTime \leftarrow rushHour, raining \quad (2)$$

Rules, in this case, are of the form *head*$\leftarrow$ *body*, where *head* is an atomic proposition or its classical negation (i.e. the head is a single literal) and *body* is a set of literals. The first rule above has an empty body, which means that the head is assumed to be unconditionally true. Such rules are called facts. The second rule encodes that whenever we can derive that it is rush hour and that it is raining, we should expect a long driving time to a particular destination. Now consider the following additional rule:

$$shortExpectedDrivingTime \leftarrow offPeak, dry, not\ accident$$

which encodes that whenever the current time is off–peak and the weather is dry we should expect a short driving time, unless we found out that there has been an accident. Since in practical situations, we will never be able to prove that there has been no accident, the not in the rule above is interpreted as negation-as-failure: if we cannot prove that there has been an accident, we assume that there has been no accident. The conclusions of an answer set program, i.e. the set of literals that we can derive to be true, is called an answer set. For programs without negation-as-failure, there is a unique answer set which corresponds to the minimal model of the program (w.r.t. set inclusion), interpreting rules as material implication. For programs with negation-as-failure there may be several alternatives (see below).

Fuzzy answer set programming (FASP) is a generalization of ASP based on fuzzy logic (Van Nieuwenborgh, De Cock, and Vermeir 2007). Essentially, literals are then associated with a truth degree from the unit interval $[0, 1]$, reflecting the intensity of a certain property. The example above, for instance, refers to properties that are a matter of degree. If it is only raining a little bit, driving time will not be substantially affected, i.e. what (2) really means is

> "If the intensity of the rush hour and the amount of rain are above a certain threshold, then driving time will be above a certain threshold."

In FASP, the need for such thresholds disappears, using rules of the form

$$longExpectedDrivingTime \leftarrow f(rushHour, raining) \quad (3)$$

where f is a particular $[0, 1]^2 \rightarrow [0, 1]$ function that encodes the exact relationship between the intensity of the rush hour, the amount of rain and the expected driving time. Different variants of FASP are obtained by restricting the class of functions f. For instance, it is common to assume that these functional relationships can be encoded using connectives

[*]Funded by a joint FWO project
[†]Postdoctoral fellow of the FWO
[‡]Funded by a joint FWO project
[§]On leave from Universiteit Gent

from particular fuzzy logics. In (Janssen et al. 2008), for instance, it is shown how answer sets of a FASP program can be found using mixed integer programming when using connectives from Łukasiewicz logic. One of the strengths of FASP lies in its ability to encode continuous optimization problems in a way which is very similar to the way discrete optimization problems are encoded in ASP. For instance, (Schockaert et al. 2009) shows how FASP can be used to find the strong Nash equilibria of games with continuous strategies, while (Janssen et al. 2010) introduces a FASP program to solve the reviewer assignment problem. However, in contrast to ASP, FASP does not appear to be suitable as a basis for commonsense reasoning. Indeed, when looking at (3), for instance, without extensive domain knowledge, we are unlikely to find a reasonably accurate estimate for the function f.

Despite the reference to the name "fuzzy", FASP is essentially a generalization of ASP to continuous domains, and not a form of approximate reasoning. As such, FASP is not well-equipped to deal with vagueness, and with uncertainty in general. Other extensions of logic programming, however, have been specifically targeted at combining ASP with theories of uncertainty, such as probability theory (Lukasiewicz 1998), possibility theory (Dubois, Lang, and Prade 1991; Nicolas et al. 2006) or evidence theory (Wan 2009), but, since they are based on classical ASP, these existing approaches can only be used in discrete domains. The aim of this paper is to combine both directions of work and develop a possibilistic extension of fuzzy answer set programming. Although combinations of FASP with other theories of uncertainty may be envisaged, the choice of possibility theory appears to be the most natural one. Indeed, as already noted by Zadeh in (Zadeh 1978), there are strong links between fuzziness and possibility, which form an important basis for theories of approximate reasoning (Dubois and Prade 1999). Similarly, as we show below, by combining FASP with possibility theory, an expressive framework is obtained that combines the advantages of ASP with those of approximate reasoning (e.g. robustness). Finally, note that being one of the simplest non-trivial theories of uncertainty, possibility theory is also an interesting choice from a computational point of view. Indeed, as it turns out, answer sets of possibilistic FASP programs can straightforwardly be found by calculating the answer sets of a small number of "regular" FASP programs.

The remainder of this paper is structured as follows. In the following section, we provide a motivating example to illustrate the need for a possibilistic extension of FASP. Subsequently, we present some background on ASP and FASP, after which we introduce possibilistic fuzzy answer set programming. We then illustrate the approximate reasoning capabilities of possibilistic FASP. Finally, we provide an overview of related work, and present some conclusions and directions for future work.

A Motivating Example

(Nowak 2006) Cancer is an evolutionary process in which cells of the body mutate and start to exhibit improper behaviour such as cells starting to procreate without boundaries, leading to malignant tumors. Luckily, multi-cellular organisms have a range of defenses to protect against the development of cancer. One line of defense are tumor suppressor genes or TSGs which are involved in promoting apoptosis or programmed cell death when a cell is found to misbehave. Mutations during cell division may cause a cell to lose its TSGs, causing it to no longer undergo apoptosis when misbehaving. Other mutations may activate chromosomal instability or CIN. Having CIN causes cells to not divide properly, considerably speeding up the rate of mutations. CIN thus helps in deactivating TSGs and may speed up the growth rate of tumors, with larger tumors as a result. We can model this phenomenon using the following rules.

$$\textbf{1:}\ tsg(on) \leftarrow not\ tsg(off)$$
$$\textbf{1:}\ tsg(off) \leftarrow not\ tsg(on)$$
$$\textbf{1:}\ cin(on) \leftarrow not\ cin(off)$$
$$\textbf{1:}\ cin(off) \leftarrow not\ cin(on)$$
$$\textbf{1:}\ 0 \leftarrow \min(tsg(on), not\ tsg(on))$$
$$\textbf{1:}\ 0 \leftarrow \min(cin(on), not\ cin(on))$$

Intuitively, the first four rules of the program describe that TSG and CIN will either be on or off. Unless further information becomes available, two choices have thus to be made, which will lead to four different answer sets. The last two rules are constraints, which describe that the body of these rules cannot be satisfied. In this case, the last two rules ensure that *tsg(on)*, *tsg(off)*, *cin(on)* and *cin(off)* are treated as crisp literals, i.e. they can only take truth values 0 and 1 in any model of the program. The **1:** in front of the rules indicates that the certainty of these rules is maximal.

In FASP, we can describe that we have a tumor of e.g. size 0.6 or more, but it is not possible to express that a tumor will be "reasonably large", which we can do in our framework. The next set of rules encode that when *cin(on)* and *tsg(off)* hold, it is likely to grow a reasonably large tumor.

$$\textbf{0.8:}\ tumor \leftarrow 0.4 \otimes cin(on) \otimes tsg(off)$$
$$\textbf{0.6:}\ tumor \leftarrow 0.6 \otimes cin(on) \otimes tsg(off)$$
$$\textbf{0.4:}\ tumor \leftarrow 0.8 \otimes cin(on) \otimes tsg(off)$$
$$\textbf{0.2:}\ tumor \leftarrow 1 \otimes cin(on) \otimes tsg(off)$$

In these rules, $\otimes$ denotes a t-norm (see the preliminaries). The truth degree of $tumor$ should be interpreted as a measure of the size of the tumor. Intuitively, when *cin(on)* and *tsg(off)* have truth degree 1, we can derive that $tumor$ has truth degree 0.4 with certainty 0.8, 0.6 with certainty 0.6, 0.8 with certainty 0.4 and 1 with certainty 0.2. Hence, it is considered somewhat plausible that a large tumor will grow, and very likely that at least a moderately-sized tumor will grow. The information encoded in our example is that the tumor will be "reasonably large", which is an elastic constraint; it is around 0.6 with the actual size between 0.4 and 1. However, we are less certain when we stretch the constraint further away from 0.6, e.g. towards 1. A similar set of rules can be added to describe inferences in cases where *tsg(off)*

holds, but not necessarily *cin(on)*:

$$\textbf{0.8: } tumor \leftarrow 0.1 \otimes tsg(off)$$
$$\textbf{0.6: } tumor \leftarrow 0.2 \otimes tsg(off)$$
$$\textbf{0.4: } tumor \leftarrow 0.3 \otimes tsg(off)$$
$$\textbf{0.2: } tumor \leftarrow 0.4 \otimes tsg(off)$$

In this case, we still consider it likely that a tumor will grow, but to a lesser extent, i.e. of a smaller size. Finally, we also consider the following rule:

$$\textbf{0.6: } tsg(off) \leftarrow cin(on)$$

This rule specifies that when *cin(on)* holds, it is likely that *tsg(off)* will be the case, i.e. CIN acts as a catalyst towards disabling TSG.

Preliminaries

Answer set programs

Let $\mathcal{A}$ be a finite set of atoms. A literal is either an atom $a \in \mathcal{A}$ or the negation $\neg a$ of an atom. We write $\mathcal{L}$ for the set of all literals. A set of literals I is called an interpretation if it is consistent, i.e. if there is no atom a such that both $a \in I$ and $\neg a \in I$. A *normal rule* is an expression of the form $l_0 \leftarrow l_1, ..., l_s, not\ l_{s+1}, ..., not\ l_n$ where $l_0, l_1, ..., l_n$ are literals. If no occurrences of not appear in a rule (i.e. $s = n$), it is called simple. A *normal (resp. simple) program* P is a finite set of normal (resp. simple) rules. An interpretation I is a *model* of a simple rule $r = (l_0 \leftarrow l_1, ..., l_s)$, denoted $I \models r$, if $l_0 \in I$ or $\{l_1, ..., l_n\} \not\subseteq I$. An interpretation I of a simple program P is a *model* of P iff $\forall r \in P \cdot I \models r$.

Answer sets are defined using the *immediate consequence operator* T_P of a simple program P, defined for a set of literals I as:

$$T_P(I) = I \cup \{l_0 \mid (l_0 \leftarrow l_1, ..., l_s) \in P \wedge (\{l_1, ..., l_s\} \subseteq I)\}$$

We use $P^\star$ to denote the fixpoint which is obtained by repeatedly applying T_P starting from the empty interpretation, i.e. the least fixpoint of T_P w.r.t. set inclusion. An interpretation I is called an answer set of a simple program P iff $I = P^\star$. Answer sets of normal programs are defined using the Gelfond-Lifschitz reduct P^I of a normal program P and interpretation I, defined as

$$P^I = \{l_0 \leftarrow l_1, ..., l_s \mid (\forall i \in \{s+1, ..., n\} \cdot l_i \notin I) \wedge (l_0 \leftarrow l_1, ..., l_s, not\ l_{s+1}, ..., not\ l_n) \in P\}$$

An interpretation I is then called an answer set of the normal program P iff $\left(P^I\right)^\star = I$, i.e. if I is the answer set of the reduct P^I.

Fuzzy answer set programs

A fuzzy rule is a rule of the form $l_0 \leftarrow f(l_1, \ldots, l_s; l_{s+1}, \ldots, l_n)$ with $l_0, \ldots, l_n \in \mathcal{L}$ and f a computable function such that the first s arguments of f are increasing and the remaining arguments are decreasing. A fuzzy program P is a set of fuzzy rules. A fuzzy simple rule is a fuzzy rule $l_0 \leftarrow f(l_1, \ldots, l_s;)$ where every argument of f is increasing. A fuzzy (simple) program P is a set of fuzzy (simple) rules. A fuzzy interpretation I is a consistent mapping $I : \mathcal{L} \rightarrow [0, 1]$, i.e. it associates with each literal a degree of truth with $I(a) + I(\neg a) \leq 1$ for all $a \in \mathcal{A}$. Furthermore, we define $[f(l_1, \ldots, l_s; l_{s+1}, \ldots, l_n)]_I = f(I(l_1), \ldots, I(l_s); I(l_{s+1}), \ldots, I(l_n))$. A fuzzy interpretation I is a model of a fuzzy simple rule $r = l_0 \leftarrow f(l_1, \ldots, l_s;)$, denoted $I \models r$ if $I(l_0) \geq [f(l_1, \ldots, l_s;)]_I$. A fuzzy interpretation I is a model of a fuzzy simple program if $I \models r$ for all $r \in P$. A fuzzy interpretation I is an answer set of P iff I is a minimal model of P. Equivalently, it is the least fixpoint of the immediate consequence operator T_P defined by (Janssen et al. 2009).

$$T_P(I)(l_0) = \sup\{[f(l_1, ..., l_s;)]_I \mid l_0 \leftarrow f(l_1, ..., l_s;) \in P\}$$

Answer sets of arbitrary fuzzy programs are defined using a generalization of the Gelfond-Lifschitz reduct, defined by

$$P^I = \{l_0 \leftarrow f(l_1, \ldots, l_s; I(l_{s+1}), ..., I(l_n)) \mid l_0 \leftarrow f(l_1, \ldots, l_s; l_{s+1}, \ldots, l_n) \in P\}$$

A fuzzy interpretation I is then an answer set of P iff M is an answer set of P^I.

Note that the definition of a fuzzy rule allows a large class of functions. In practice, however, it is common to define fuzzy rules using connectives from a particular fuzzy logic. For instance, rules can then be of the form $a \leftarrow b \otimes (c \oplus (1 - d))$ where $\otimes$ and $\oplus$ are respectively a t-norm and a t-conorm. Recall that a t-norm is a commutative, associative, monotonically increasing $[0, 1]^2 \rightarrow [0, 1]$ mapping $\otimes$ satisfying $1 \otimes x = x$ for all $x \in [0, 1]$, while a t-conorm is a commutative, associative, monotonically increasing $[0, 1]^2 \rightarrow [0, 1]$ mapping $\oplus$ satisfying $0 \oplus x = x$ for all $x \in [0, 1]$. In particular, the minimum and maximum are a well-known t-norm and t-conorm, respectively, which we write as $\wedge$ and $\vee$. While there is no a priori theoretical benefit from restricting the class of functions to those that can be defined in terms of fuzzy logic connectives, this restriction makes fuzzy programs often easier to understand.

Possibilistic Fuzzy Answer Set Programming

In (Nicolas et al. 2006; Nieves, Osorio, and Cortés 2007) the semantics of answer set programs are extended with possibility theory to arrive at a new framework that is able to deal with reasoning that is uncertain and non-monotonic. In such a setting, a necessity degree is associated with each literal and rule, allowing to rank information in terms of its certainty. We now introduce a similar extension to the semantics of fuzzy answer set programs.

Language

Let $\mathcal{L}$ be a set of literals. In the remainder of this paper, we let C be a finite set of certainty degrees, with $C \subseteq]0, 1]$. We define a valuation as a function $V : \mathcal{L} \rightarrow (C \rightarrow [0, 1])$ which maps each literal to a decreasing function. The intuition is that for a literal l, $V(l)(c) = d$ means that we can derive with certainty c that the truth degree of l is at least d. For increasing certainty values, the lower bound on the truth value of l will be weaker, i.e. lower. For notational

convenience we also use the set notation $V = \{l^{c;d}, \ldots\}$ to specify valuations in which $V(l)(c) = d$. Given two valuations V and V', we define $V \leq V'$ as the pointwise extension of the natural ordering on the truth degrees, i.e. $\forall l \in \mathcal{L} \cdot \forall c \in C \cdot V(l)(c) \leq V'(l)(c)$. For $c \in C$ and a valuation V, we let V^c be the fuzzy interpretation defined by $V^c(l) = V(l)(c)$. We say that a valuation V is *c-consistent* whenever V^c is consistent. A valuation V is said to be *consistent* when it is c_0-consistent with $c_0 = min(C)$. A (possibilistic fuzzy) interpretation I is a consistent valuation.

Example 1. *Consider the valuations* $V = \{a^{1;0.6}, b^{1;0.4}\}$ *and* $V' = \{a^{1;0.6}, b^{1;0.6}, \neg a^{0.8;0.5}\}$ *with* $C = \{1, 0.8\}$. *We have that* $V(b)(1) = 0.4 \leq V'(b)(1) = 0.6$. *A similar observation can be made for* a *and* $\neg a$ *for every* c *and thus* $V \leq V'$. *We have that* V' *is not consistent; with* $min(C) = 0.8$ *we get* $V'(a)(0.8) = 0.6$ *(since* $V'(a)(1) = 0.6$ *and the function is decreasing) and* $V'(\neg a)(0.8) = 0.5$ *which is not consistent since* $0.6 + 0.5 \not\leq 1$. V *however is consistent.*

A *possibilistic fuzzy simple program* is a set of *possibilistic fuzzy simple rules*. A possibilistic fuzzy simple rule is a pair $p = (r, n(r))$ with r a fuzzy simple rule and $n(r)$ a certainty value associated with r. We write a pair $p = (r, n(r))$ with $r = (l_0 \leftarrow f(l_1, \ldots, l_s;))$ as:

$$\textbf{n(r):}\ l_0 \leftarrow f(l_1, \ldots, l_s;).$$

The c-cut P_c of a possibilistic fuzzy simple program P is then defined as

$$P_c = \{(r, n(r)) \in P \mid n(r) \geq c\}.$$

Example 2. *We know with certainty that it is quite cold outside. It is quite likely that when the streets are wet and temperatures are below zero, the streets are wet because of snow rather than rain. We are certain that when it is cold and snowing, it will be very risky on the roads. It is possible that when it is cold, it will be a bit risky and when it is wet, that the roads will be rather risky. We have the program* P *with the rules:*

$$\begin{aligned}
\textbf{1:}\ & cold \leftarrow 0.6 \\
\textbf{1:}\ & wet \leftarrow 0.4 \\
\textbf{1:}\ & risky \leftarrow cold \cdot snow \\
\textbf{0.8:}\ & snow \leftarrow (cold \geq 0.5) \wedge wet \\
\textbf{0.6:}\ & risky \leftarrow 0.5 \cdot cold \\
\textbf{0.6:}\ & risky \leftarrow 0.8 \cdot wet.
\end{aligned}$$

The first two rules express facts about how cold and wet it is, respectively. The third rule states that the degree of risk of the roads is proportional to the degree of wetness and coldness. The fourth rule states that the degree of wetness corresponds to the degree to which there is snow, provided that it is cold at least to degree 0.5, which we assume to correspond to $0°C$. *The last two rules allow us to derive some weaker information about the risk even when there is no snow, though with only a limited certainty. We have that* P_1 *contains the first three rules,* $P_{0.8}$ *contains the first four rules and* $P_{0.6} = P$.

Fixpoint theory

Definition 1. *Let* P *be a possibilistic fuzzy simple program and* X *a valuation. The immediate consequence operator* T_P *is defined for* $l \in \mathcal{L}$ *and* $c \in C$ *as:*

$$T_P(X)(l)(c) = \sup\{[\alpha]_{X^c} \mid (r : l \leftarrow \alpha) \in P_c\}$$

where we assume $\sup \emptyset = 0$.

Given a valuation, the immediate consequence operator determines the highest truth degree that can be motivated by the rules, at a given certainty level c, by applying the rules in P once on what has already been established. Due to a result by Tarski (Tarski 1955) and because our consequence operator T_P is monotonic, we know that our consequence operator has a least fixpoint, denoted as $P^\star$ for any possibilistic fuzzy simple program P. This least fixpoint can be computed using an iterated fixpoint procedure, i.e. by applying T_P repeatedly, starting from the minimal interpretation $\emptyset$.

Example 3. *Consider* P *from Example 2. We have:*

$$\begin{aligned}
T_P(\emptyset) = S_0 = \ & \{cold^{1;0.6}, wet^{1;0.4}\} \\
T_P(S_0) = S_1 = \ & \{cold^{1;0.6}, wet^{1;0.4}, snow^{0.8;0.4}, \\
& risky^{0.6;0.32}\} \\
T_P(S_1) = P^\star = \ & \{cold^{1;0.6}, wet^{1;0.4}, snow^{0.8;0.4}, \\
& risky^{0.6;0.32}, risky^{0.8;0.24}\}
\end{aligned}$$

Proposition 1. T_P *is monotonic. That is, given two interpretations* I *and* I' *such that* $I \leq I'$, *it holds that* $T_P(I) \leq T_P(I')$.

Definition 2. *Let* P *be a possibilistic fuzzy simple program. An interpretation* I *is an answer set of* P *iff* $I = P^\star$.

Note how the immediate consequence operator from Definition 1 is a generalization of the immediate consequence operator that is used for fuzzy simple programs. When we take C such that $|C| = 1$, i.e. we only consider a single certainty level, then our immediate consequence operator coincides with the one from fuzzy answer set programming, when ignoring the certainty values. In other words, we can interpret any fuzzy simple program as a possibilistic fuzzy simple program in which every rule is certain.

Model theory

We have thus far provided a syntactical deduction process based on a fixpoint operator defined on rules of a possibilistic fuzzy simple program. Despite being a natural way to define answer sets, it does not readily lead to a way to actually implement a solver for possibilistic fuzzy answer set programs. This is due to the fact that for some programs the fixpoint of the immediate consequence operator is only obtained after infinitely many applications. In this section, we provide a characterization of anwer sets in terms of minimal models, which will offer us a way to find answer sets of possibilistic fuzzy simple programs using existing fuzzy answer set solvers (Janssen et al. 2008).

Definition 3. *An interpretation* I *is a model of a possibilistic fuzzy simple rule of the form* $r =$ **c:** $l \leftarrow body$, *denoted as* $I \models r$ *iff* $I(l)(c) \geq [body]_{I^c}$.

Definition 4. *An interpretation I is a model of a possibilistic fuzzy simple program P iff $\forall r \in P \cdot I \models r$.*

Using the c-cuts of a possibilistic fuzzy simple program, we can look at a possibilistic fuzzy program as a layered program where higher layers correspond to more certain rules. As we show next, a possibilistic fuzzy simple program has a unique minimal model which corresponds to the set of minimal models of the fuzzy simple programs P_c for each $c \in C$.

Proposition 2. *Let P be a possibilistic fuzzy simple program. Let the valuation M be defined by $\forall l \in \mathcal{L} \cdot (\forall c \in C \cdot M(l)(c) = F_c(l))$ with F_c the unique minimal model of the fuzzy simple program P_c. Then M is the minimal model of P.*

Proof. We prove this by contradiction. Let M' be a minimal model and assume that $M' \neq M$. Since M is not the minimal model we have that $\exists l \in \mathcal{L} \cdot \exists c \in C \cdot (M(l)(c) > M'(l)(c))$. Let us define $F'_c = (M')^c$; note that we also have $F_c = (M')^c$. We can then rewrite the condition that M is not a minimal model as $\exists l \in \mathcal{L} \cdot F'_c(l) < F_c(l)$. Since F_c is the minimal model of P_c, this is only possible if F'_c is not a model of P_c. Hence $\exists l' \in \mathcal{L} \cdot (\exists((l' \leftarrow \alpha') \in P_c) \cdot [\alpha']_{F'_c} > F'_c(l'))$. Because of the definition of P_c we know that there is a rule c'**:** $l' \leftarrow \alpha' \in P$ with $c' \geq c$. However, we have just shown that M' is not a model of this rule, thus M' cannot be a model of P, and a fortiori not a minimal model. □

If we take a closer look at the consequence operator from Definition 1, it is easy to see that it works by taking the c-cut and then computing the answer set of the fuzzy projection of the possibilistic fuzzy simple program.

Corollary 1. *Let P be a possibilistic fuzzy simple program. Then $P^\star$ is the unique minimal model of P.*

Syntactic Extensions

Approximate conditions

In this section we introduce a number of syntactic extensions that allow us to write possibilistic fuzzy answer set programs more succinctly and often more naturally. The first extension allows us to deal with approximate information in the body. Specifically, this extension expresses that we do not have full confidence w.r.t. some information in the body of a rule. In such a case, we may either weaken the information in the body or lower the certainty of the conclusion, or both.

Definition 5. *Let $f : C \to [0,1]$ and $\otimes : [0,1]^2 \to [0,1]$ be functions. We define the approximation rule $l \leftarrow approx^f_\otimes(body)$ as a shorthand for the set of rules $\{$**c:** $l \leftarrow body \otimes f(c) \mid c \in C\}$.*

Example 4. *Consider the program with the rule*

$$happy \leftarrow approx^f_\otimes(warm \otimes sunny)$$

with $f(x) = 1.2 - x$, $\otimes$ the product operator, and $C = \{0.2, 0.4, ..., 1\}$. This is a shorthand for

$$\textbf{1:}\ happy \leftarrow warm \otimes sunny \otimes 0.2$$
$$\vdots$$
$$\textbf{0.2:}\ happy \leftarrow warm \otimes sunny \otimes 1.$$

i.e. when it is warm and sunny, we are definitely happy to some extent. With a limited certainty, we can even derive that we are completely happy in such a case. Now consider also the following rules:

$$\textbf{1:}\ warm \leftarrow 0.8$$
$$\textbf{1:}\ sunny \leftarrow 0.6$$

We have

$$\{warm^{0.8;1}, sunny^{0.6;1}, happy^{0.48;0.2}, happy^{0.384;0.4}$$
$$happy^{0.288;0.6}, happy^{0.192;0.4}, happy^{0.096;0.4}\}$$

as the answer set of this program.

An important special case of this general idea are rules that derive certainty information about the truth value of some atom a based on a possibility distribution π_a (i.e. a $[0,1] \to [0,1]$ mapping from truth values to possibility degrees that restricts the possible values of a). Let a_c for $c \in C$ be defined as $a_c = \max\{x_0 \in [0,1] | \inf_{x<x_0}(1 - \pi_a(x)) \geq c\}$, then possibility theory dictates that a_c is the strongest lower bound that we can establish for a with certainty c (in terms of the necessity measure[1] induced by π_a). We write $l \leftarrow approx^{\pi_a}$ for the set of rules $\{$**c:** $a \leftarrow a_c | c \in C\}$.

Variable certainty weights

Rules in fuzzy answer set programming are truth-qualifying, i.e. the intuition is that the more the body is true, the more the head of the rule should be true. Another type of fuzzy rules are uncertainty-qualifying rules (Dubois and Prade 1996) that model the intuition that the more the body of a rule is true, the more certain that the head is true. In our framework, this corresponds to possibilistic fuzzy rules in which the certainty score is an expression involving literals, rather than a constant. Specifically, we will consider rules of the form:

$$\mathbf{f_1(l_1, ..., l_n)}\textbf{:}\ l \leftarrow f_2(l'_1, ..., l'_m)$$

Such rules can easily be simulated using the framework introduced above using the following set of rules:

$$\{\textbf{c:}\ l \leftarrow f_2(l'_1, ..., l'_m) \wedge (f_1(l_1, ..., l_n) \geq c) | c \in C\}$$

where for a valuation I, $[f_1(l_1, ..., l_n) \geq c]_I = 1$ if $[f_1(l_1, ..., l_n)]_I \geq c$ and $[f_1(l_1, ..., l_n) \geq c]_I = 0$ otherwise.

Example 5. *(modified from (Janssen et al. 2009)) Consider a number of people trying to decide whether they should organize a barbecue. The certainty that they will indeed want a barbecue is determined by the degree of sunshine, whereas the size of the barbecue is determined by the appetite of the participants. We write:*

$$\textbf{sunshine:}\ bbq \leftarrow hungry$$

For $C = \{0.2, 0.4, \ldots, 1\}$ this corresponds to the rules

$$\textbf{0.2:}\ bbq \leftarrow hungry \otimes (sunshine \geq 0.2)$$
$$\textbf{0.4:}\ bbq \leftarrow hungry \otimes (sunshine \geq 0.4)$$
$$\vdots$$
$$\textbf{1:}\ bbq \leftarrow hungry \otimes (sunshine \geq 1).$$

[1] Recall that given a possibility distribution π in a universe U, the possibility $\Pi(A)$ and necessity $N(A)$ of a set $A \subseteq U$ are defined as $\Pi(A) = \sup_{u \in A} \pi(u)$ and $N(A) = 1 - \Pi(U \setminus A)$.

Add the following facts:

$$\mathbf{1:}\ sunshine \leftarrow 0.9$$
$$\mathbf{1:}\ hungry \leftarrow 0.2.$$

The resulting answer set is

$$\left\{hungry^{1;0.2}, sunshine^{1;0.9}, bbq^{0.8;0.2}\right\}.$$

That is, it is likely that we will have a barbecue, but the barbecue will be rather small.

Approximate Reasoning

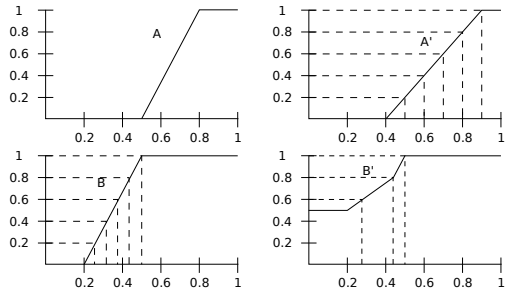

Figure 1: Possibility Distributions

In this section we show how approximate reasoning with if-then rules under generalized modus ponens (GMP) can be supported in possibilistic fuzzy answer set programming. The idea of GMP is that, given information about the possible values of a variable x in the form of a possibility distribution A' (usually written as "x is A'"), we can derive possible values for a variable y (usually written as "y is B'") from a rule of the form "if x is A then y is B", where A and B are fuzzy sets in the universes U and V corresponding to variables x and y. To illustrate the main ideas, we will consider the possibility distribution on y defined by $B'(v) = \sup_u \min(A'(u), A(u) \rightarrow B(v))$ for all $v \in V$, where $\rightarrow$ corresponds to the Łukasiewicz implicator[2] and A', A and B are as depicted in Figure 1. The resulting possibility distribution B' is also shown.

To find an approximation of B' using possibilistic fuzzy answer set programs, the information encoded in the fuzzy sets/possibility distributions A', A and B is discretized by only considering the certainty levels from $C = \{0.2, 0.4, 0.6, 0.8, 1\}$. For example, we can derive from "x is A'" that $x \geq 0.5$ with certainty 0.8 since $N(\{x|x \geq 0.5\}) = 1 - \Pi(\{x|x < 0.5\}) = 1 - A'(0.5) = 0.8$. The information x *is* A' can thus be encoded by the rule:

$$x \leftarrow approx^{A'} \tag{4}$$

Intuitively, this rule states that we may conclude with certainty c that x is not smaller than the infimum of the strict c-cut $A'_{\underline{c}} = \{x|A(x) > c\}$. Figure 1 illustrates these c-cuts and corresponding infima for $c \in C$ with dashed lines. From this we can see that rule (4) corresponds to the following set:

$$\mathbf{1:}\ x \leftarrow 0.4 \quad \mathbf{0.8:}\ x \leftarrow 0.5 \quad \mathbf{0.6:}\ x \leftarrow 0.6$$
$$\mathbf{0.4:}\ x \leftarrow 0.7 \quad \mathbf{0.2:}\ x \leftarrow 0.8 \quad \mathbf{0:}\ x \leftarrow 0.9$$

[2]Recall that the Łukasiewicz implicator is defined by $a \rightarrow b = \min(1, 1 - a + b)$ for all $a, b \in [0, 1]$

The information that "if x is A then y is B" essentially corresponds to a conditional possibility distribution on y, depending on the value of x. The certainty that $y \geq v$ for a certain $v \in V$ is given by $N(\{y|y \geq v\}) = 1 - \sup(\{A(x) \rightarrow B(y)|y < v\}) = 1 - (A(x) \rightarrow B(v))$. Writing $b_c = \inf B_{\underline{c}}$ for the lower bound of the c-cut of B, we thus find:

$$\{\mathbf{1} - (\mathbf{A}(\mathbf{x}) \rightarrow \mathbf{c}) : y \leftarrow b_c \mid c \in C\}$$

Note how the fact that the possibility that y takes on a certain value is conditional on the value of x, leads to variable certainty weights. As an example, let us consider the rules for certainty level $c = 0.4$:

$$\mathbf{0.4:}\ y \leftarrow 0.2 \wedge (1 - (A(x) \rightarrow 0) \geq 0.4)$$
$$\mathbf{0.4:}\ y \leftarrow 0.26 \wedge (1 - (A(x) \rightarrow 0.2) \geq 0.4)$$
$$\mathbf{0.4:}\ y \leftarrow 0.32 \wedge (1 - (A(x) \rightarrow 0.4) \geq 0.4)$$
$$\mathbf{0.4:}\ y \leftarrow 0.38 \wedge (1 - (A(x) \rightarrow 0.6) \geq 0.4)$$
$$\mathbf{0.4:}\ y \leftarrow 0.44 \wedge (1 - (A(x) \rightarrow 0.8) \geq 0.4)$$
$$\mathbf{0.4:}\ y \leftarrow 0.5 \wedge (1 - (A(x) \rightarrow 1) \geq 0.4)$$

If we want to find a lower bound for the value of y that can be derived with certainty c, we need to calculate the classical answer set of the program obtained by removing all rules with certainty strictly below c. This leads to the following values (lower bounds) of y: $y = 0.26$ with certainty 0.6, $y = 0.44$ with certainty 0.8, and $y = 0.5$ with certainty 1. As one can see from Figure 1, the resulting output is an approximation of the conclusions that are obtained using the GMP, in the sense that slightly more informative conclusions (i.e. higher lower bounds) could be obtained by considering more certainty values.

Related Work

In the last few years, the combination of many-valued logic with logic programming has received a lot of attention. In (Van Nieuwenborgh, De Cock, and Vermeir 2007) the stable model semantics (Gelfond and Lifzchitz 1988) are extended to allow answer sets where literals are true to a certain degree. In (Straccia 2006; Lukasiewicz and Straccia 2007; Damásio, Medina, and Ojeda-Aciego 2004; Lakshmanan and Shiri 2001) general expressions are used as rule bodies with many-valued predicates as arguments. (Janssen et al. 2009) proposes a general framework for dealing with fuzzy answer set programs.

A large body of research has also been devoted to logic programming with uncertainty. One of the most widely spread forms of representing uncertain information attaches probabilities to statements, though this method has been repeatedly criticized (McCarthy and Hayes 1969; Dubois and Prade 2004). A more elegant approach comes in the form of possibilistic logic (Dubois, Lang, and Prade 1991), where an uncertainty degree is attached to each statement.

(Nicolas et al. 2006) proposed a framework that combines possibility theory with the stable model semantics (Gelfond and Lifzchitz 1988). Such a framework can deal with non-monotonicity and uncertainty at the same time. (Alsinet et al. 2008) shows an alternative to our approach, though for defeasible logic programming, where atoms are moreover interpreted as fuzzy sets of truth values.

Note that approximate reasoning has also been embedded and studied in other logical frameworks, for example in (Bobillo and Straccia 2008) Mamdani-style inference is embedded in Fuzzy Description Logics, and in (Perfilieva 2006) generalized modus ponens is studied in the logic of BL.

Conclusions

We have introduced possibilistic fuzzy answer set programming as an extension of FASP programs where we associate a necessity degree with each fuzzy literal and fuzzy rule. We defined a consequence operator for possibilistic fuzzy simple programs and proved that the least fixpoint of this operator corresponds with the unique minimal model of that program. This minimal model can moreover be found by calculating the answer sets of the fuzzy simple programs at each certainty level. This observation allows to readily implement our new framework on existing solvers for (general) FASP. With the aim of writing programs in a more natural way, we introduced an approximation operator and rules whose certainty depends on the truth value of particular literals. We illustrated how these extensions can be used to provide an elegant representation of approximate reasoning under generalized modus ponens.

In this paper, we omitted a discussion of negation-as-failure. The addition of negation-as-failure to possibilistic fuzzy answer set programs, however, offers no additional difficulties over those that are already encountered in (crisp) possibilistic answer set programming. Nonetheless, the issue of what the semantics of negation-as-failure should be in the presence of necessity values is an interesting issue, which will be addressed in more detail in future work. Another interesting issue is how partially satisfied answer sets from FASP and c-cuts from possibilistic ASP should complement each other when trying to resolve inconsistencies in answer set programs.

References

Alsinet, T.; evar I., C.; Godoís, L.; Sandri, S.; and Simari, G. 2008. Formalizing argumentative reasoning in a possibilistic logic programming setting with fuzzy unification. *Int. J. Approx. Reasoning* 48(3):711–729.

Bobillo, F., and Straccia, U. 2008. fuzzyDL: An expressive fuzzy description logic reasoner. In *Proc. of FUZZ-08*, 923–930. IEEE Computer Society.

Damásio, C. V.; Medina, J.; and Ojeda-Aciego, M. 2004. Sorted multi-adjoint logic programs: Termination results and applications. In *JELIA2004*, 252–265.

Dubois, D., and Prade, H. 1996. What are fuzzy rules and how to use them. *Fuzzy Sets and Systems* 84(2):169–185.

Dubois, D., and Prade, H. 1999. Fuzzy sets in approximate reasoning, part 1: inference with possibility distributions. *Fuzzy Sets and Systems* 100:73–132.

Dubois, D., and Prade, H. 2004. Possibilistic logic: a retrospective and prospective view. *Fuzzy Sets and Systems* 144(1):3–23.

Dubois, D.; Lang, J.; and Prade, H. 1991. Towards possibilistic logic programming. In *Proc. of ICLP'91*, 581–595.

Gelfond, M., and Lifzchitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, 1081–1086. Seattle, USA: ALP, IEEE.

Janssen, J.; Heymans, S.; Vermeir, D.; and Cock, M. D. 2008. Compiling fuzzy answer set programs to fuzzy propositional theories. In *Proc of ICLP'08*, 362–376.

Janssen, J.; Schockaert, S.; Vermeir, D.; and Cock, M. 2009. General fuzzy answer set programs. In *Proc. of WILF'09*, 352–359.

Janssen, J.; Schockaert, S.; Vermeir, D.; and De Cock, M. 2010. General fuzzy answer set programming: the basic language. Submitted.

Lakshmanan, L. V., and Shiri, N. 2001. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering* 13:554–570.

Lukasiewicz, T., and Straccia, U. 2007. Tightly integrated fuzzy description logic programs under the answer semantics for the semantic web. In *Proc. of RR-07*, 289—298.

Lukasiewicz, T. 1998. Probabilistic logic programming. In *Proc. of ECAI'98*, 388–392.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*. Edinburgh University Press. 463–502.

Nicolas, P.; Garcia, L.; Stéphan, I.; and Lefèvre, C. 2006. Possibilistic uncertainty handling for answer set programming. *Ann Math Artif Intell* 47(1-2):139–181.

Nieves, J. C.; Osorio, M.; and Cortés, U. 2007. Semantics for possibilistic disjunctive programs. In *Proc. of LPNMR*, 315–320.

Nowak, M. 2006. *Evolutionary Dynamics: Exploring the Equations of Life*. Belknap Press of Harvard University Press.

Perfilieva, I. 2006. *Computational Intelligence, Theory and Applications*. Springer–Verlag. chapter Fuzzy IF-THEN Rules from Logical Point of View.

Schockaert, S.; Janssen, J.; Vermeir, D.; and De Cock, M. 2009. Answer sets in a fuzzy equilibrium logic. In *Proc. of RR2009*, 135–149.

Straccia, U. 2006. Annotated answer set programming. In *Proc. of IPMU-06*, 1212–1219.

Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5:285–309.

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

Wan, H. 2009. Belief logic programming. In *Proc. of ICLP'09*, 547–548.

Zadeh, L. A. 1978. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* 3–28.