Possible and Necessary Answer Sets of Possibilistic Answer Set Programs

Bauters, K., Schockaert, S., De Cock, M., & Vermeir, D. (2012). Possible and Necessary Answer Sets of Possibilistic Answer Set Programs. In *Proceedings of the 24th International Conference on Tools with Artificial Intelligence (ICTAI'12)* (pp. 836-843). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICTAI.2012.117

Published in:
Proceedings of the 24th International Conference on Tools with Artificial Intelligence (ICTAI'12)

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Possible and Necessary Answer Sets of Possibilistic Answer Set Programs

Kim Bauters*, Steven Schockaert†, Martine De Cock* and Dirk Vermeir‡
*Department of Applied Mathematics and Computer Science
Universiteit Gent, Krijgslaan 281, 9000 Gent, Belgium
Email: (kim.bauters, martine.decock)@ugent.be
†School of Computer Science & Informatics, Cardiff University
5 The Parade, Cardiff CF24 3AA, United Kingdom
Email: s.schockaert@cs.cardiff.ac.uk
‡Department of Computer Science
Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium
Email: dvermeir@vub.ac.be

***Abstract*—Answer set programming (ASP) and possibility theory can be combined to form possibilistic answer set programming (PASP), a framework for non-monotonic reasoning under uncertainty. Existing proposals view answer sets of PASP programs as weighted epistemic states, in which the strength by which different literals are believed to hold may vary. In contrast, in this paper we propose an approach in which epistemic states remain Boolean, but some epistemic states may be considered more plausible than others. A PASP program is then a representation of an incomplete description of these epistemic states where certainties are associated with each rule which is interpreted in terms of a necessity measure. The main contribution of this paper is the introduction of a new semantics for PASP as well as a study of the resulting complexity.**

I. Introduction

Answer Set Programming (ASP) is a form of declarative programming which is based on the notion of a stable model. A rule such as ($beach \leftarrow sunny, hot$) intuitively encodes that when it is sunny and hot, we want to go to the beach. ASP also allows us to reason about incomplete information by means of negation-as-failure (NAF). The slightly altered rule ($beach \leftarrow sunny, hot, not\ crowded$) intuitively encodes that we go to the beach when it is sunny and hot, unless we have reasons to believe that it is crowded. Of particular interest is that our conclusion may change over time, i.e. we may need to revise our conclusions, as NAF is a non-monotonic operator. Indeed, if we are underway to the beach and we hear on the radio that it is crowded at the beach, we would no longer be inclined to go.

While ASP can deal with incomplete information, it does not provide a mechanism to express that some rules or facts are more certain than others. Possibilistic ASP (PASP) combines ASP with possibility theory, a popular formalism for dealing with uncertainty, by associating a certainty degree with each rule. We are then not only interested in finding out whether or not we can, for example, derive '$beach$', but also with what certainty we can derive it. For programs without NAF, the certainty with which we can derive the consequent is given by the weakest information needed to derive the conclusion. Thus it is limited by the certainty of either the rule itself or by the certainties of the components that make up the antecedent. Indeed, if we only know with little certainty that it will be hot, then we will have little certainty that we will actually go to the beach. When we consider NAF, however, there are different ways to deal with the uncertainty attached to each rule. One possibility is to rely on the classical semantics of ASP to determine the reduct (i.e. we assume that the program is a classical program, without certainties) and take the certainties into account for the resulting simple program [1]. Such an approach has the benefit of staying close to the classical semantics of ASP. Alternatively, we may also choose to interpret '$not\ a$' as the degree to which we are not able to derive the conclusion 'a' with certainty [2]. Then PASP coincides with a particular multi-valued semantics [3], although for programs that allow disjunctive consequents there may be differences [4].

This paper explores another interpretation of PASP programs in which we consider a PASP program as an incomplete specification of a classical ASP program. In other words: we express our certainty in whether or not specific rules in the program are valid and thus whether or not those rules should be taken under consideration to derive conclusions. The intuition of our semantics is then that literals can be seen as plausible consequences of a program, even if they can only be derived when a particular subset of the rules is considered. Clearly, the higher the certainty of the rules that need to be omitted, the lower the plausibility of the literals that can be derived from them. Conversely, a literal can only be considered as a necessary consequence if we can still derive it when some of the less certain rules are omitted.

As a running example, consider the following problem setting. The weather report indicates that there is a slight chance of rain ($rain$). As long as it is not raining, we are rather likely to go to the beach ($beach$) or, somewhat less likely, to have a barbecue (bbq) due to time constraints. We

Kim Bauters is funded by a joint FWO project

cannot have both, as you are not allowed to have a barbecue on the beach. This can intuitively be encoded as the program P_{ex} with the rules

$$\begin{aligned}
\mathbf{0.2}&: rain \leftarrow \\
\mathbf{0.8}&: beach \leftarrow not\ bbq, not\ rain \\
\mathbf{0.6}&: bbq \leftarrow not\ beach, not\ rain \\
\mathbf{1}&: \leftarrow bbq, beach.
\end{aligned}$$

Intuitively, considering the problem setting, we desire an answer such that it is clear that going to the beach is preferred to having a barbecue. However, as we will see in more detail throughout this paper, the semantics from [1] and [2] are unable to arrive at this conclusion.

The remainder of this paper is organized as follows. In Section II we provide the reader with some notions from answer set programming, possibility theory and possibilistic answer set programming. In Section III we propose our new semantics for possibilistic answer set programming. The complexity results of some of the main reasoning tasks of these semantics are discussed in Section IV. Related work is discussed in Section V and we formulate our conclusions in Section VI.

II. Preliminaries

We start by reviewing the definitions of answer set programming and possibility theory that will be used in the remainder of the paper. We also review existing approaches to possibilistic answer set programming.

A. Answer Set Programs (ASP)

Answer set programming is a form of declarative programming. To define the syntax of ASP [5], we start from a finite set of atoms $\mathcal{A}$. A *literal* is either an atom 'a' or its classical negation '$\neg a$'. A *naf-literal* is either a literal 'l' or a literal 'l' preceded by not, which we call the negation-as-failure (NAF) operator. Intuitively, we have that '$not\ l$' is true when we cannot prove 'l'. An expression of the form $l_0 \leftarrow l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$ with l_i a literal for $0 \leq i \leq n$ is called a *normal rule*. We call l_0 the *head* of the rule and $l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$ the *body*. Whenever a normal rule does not contain NAF, i.e. when $n = m$, we say that the rule is a *simple rule*. A rule of the form $l_0 \leftarrow$ is called a fact and is a shorthand for $l_0 \leftarrow \top$ where $\top$ denotes tautology. A rule of the form $\leftarrow l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$ is called a constraint and is a shorthand for $\bot \leftarrow l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$ where $\bot$ denotes contradiction. A normal (resp. simple) program P is a set of normal (resp. simple) rules.

The semantics of ASP are defined as follows. The set of literals that are relevant for a normal program P are defined as $\mathcal{A}_P = (\mathcal{B}_P \cup \neg\mathcal{B}_P)$ with $\mathcal{B}_P$ the set of atoms appearing in program P and $\neg\mathcal{B}_P = \{\neg a \mid a \in \mathcal{B}_P\}$. An *interpretation* I of P is any set of literals $I \subseteq \mathcal{A}_P$. A *consistent interpretation* I is an interpretation I that does not contain both a and $\neg a$ for some $a \in \mathcal{B}_P$. An interpretation I is said to be a *model of a simple rule* of the form $l_0 \leftarrow l_1, ..., l_m$ if $l_0 \in I$ or $\{l_1, ..., l_m\} \not\subseteq I$. An interpretation I of a simple program P is a *model* of P either if I is consistent and for every rule $r \in P$ we have that I is a model of r, or, if $I = \mathcal{A}_P$. We say that a model I is an *answer set* of the simple program P when I is the minimal model w.r.t. set inclusion of P. The reduct [5] P^I of a normal program P w.r.t. an interpretation I is defined as

$$\begin{aligned}
P^I = \{l_0 \leftarrow l_1, ..., l_m \mid\ & \{l_{m+1}, ..., l_n\} \cap I = \emptyset \\
& \wedge (l_0 \leftarrow l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n) \in P\}.
\end{aligned}$$

We say that I is an answer set (or stable model) of the normal program P when I is an answer set of the simple program P^I. When a program P has the answer set $\mathcal{A}_P$, then this is the unique answer set [6] and we say that P is an *inconsistent program*. Two important reasoning tasks in ASP are brave and cautious reasoning, i.e. finding whether a literal is true in at least one or in all answer sets of a program P. We use $\models^{\text{b}}$ (resp. $\models^{\text{c}}$) to denote brave (resp. cautious) inference in classical ASP.

B. Possibility Theory

Possibility theory is a formalism for dealing with incomplete and uncertain information [7]. Let x be a variable taking its value from the universe U. A possibility distribution π is an $U \rightarrow [0, 1]$ mapping that encodes for each value $u \in U$ to what extent it is possible that u is the actual value assigned to x [7]. By convention, $\pi(u) = 0$ means that u is impossible and $\pi(u) = 1$ means that no available information prevents u from being the actual value. In other words: $\pi(u)$ expresses to what degree we would not be surprised to learn that the actual value is u [8]. Possibility degrees are mainly interpreted qualitatively, i.e. when $\pi(u) > \pi(u')$, u is considered more plausible than u'. A possibility distribution π is said to be *normalized* if $\exists u \in U \cdot \pi(u) = 1$, i.e. at least one value is entirely possible. A possibility distribution π induces two uncertainty measures that allow us to rank sets of values $A \subseteq U$. The *possibility measure* Π is defined by $\Pi(A) = \max\{\pi(u) \mid u \in A\}$ and evaluates to what extent there is at least one element in A that is a possible value of x. The dual *necessity measure* N is defined by $N(A) = 1 - \Pi(U \setminus A)$ and evaluates to what extent the values outside of A are impossible (or, for normalized possibility distributions, to what extent we can be certain that x is a value in A).

C. Possibilistic ASP (PASP)

PASP combines ASP and possibility theory by associating a necessity degree with rules. We use the name PASP for a family of approaches that share a common syntax and which all rely on possibility theory. A number of semantics for PASP exist [1], [2], each with their own underlying intuition. These semantics, in general, do not agree, unless we restrict ourselves to simple programs. The semantics which we will discuss both start from a generalization of the concept of an interpretation. In classical ASP, an interpretation can be seen as a mapping $I : \mathcal{A}_P \rightarrow \{0, 1\}$, i.e. a literal $l \in \mathcal{A}_P$

is either true or false. This notion is generalized in PASP to a *valuation*, which is a function $V : \mathcal{A}_P \rightarrow [0, 1]$. The underlying intuition of $V(l) = c$ is that we can derive the literal l (or, more precisely, that l is true) with certainty 'c'. For notational convenience, the set notation $V = \{l^c, \ldots\}$ is also used. A possibilistic normal (resp. simple) program is a set of pairs $p = (r, c)$, with r a normal (resp. simple) rule and $c \in [0, 1]$ a certainty associated with r, which we will also write as $\mathbf{c} : l_0 \leftarrow l_1, ..., l_m, not\ l_{m+1}, ..., not\ l_n$. Intuitively, the certainty c attached to a rule expresses to what degree we are certain that the rule is indeed true. For a possibilistic rule $p = (r, c)$ we use p^* to denote r (i.e. the classical rule obtained by ignoring the certainty) and for a possibilistic program P we use P^* to denote the set of rules $\{p^* \mid p \in P\}$.

For simplicity, we will only present the main intuitions of the PASP semantics proposed by Nicolas et al. [1] and Bauters et al. [2]. Consider the possibilistic simple program

$$\mathbf{0.5} : a \leftarrow \qquad \mathbf{0.8} : b \leftarrow \qquad \mathbf{0.9} : a \leftarrow b.$$

The unique possibilistic answer set of this program is $\left\{a^{0.8}, b^{0.8}\right\}$. Indeed, clearly we can derive $a^{0.5}$ and $b^{0.8}$ as these are given as facts. Using the last rule, with a certainty of 0.9, we can further strengthen our conclusion. The weakest point of information, however, is that we are only able to derive 'b' with a certainty of 0.8. Hence we can only derive 'a' with a certainty of $\min(0.8, 0.9) = 0.8$.

The semantics from Nicolas [1] for a possibilistic normal program are intuitively obtained by ignoring the certainties associated with the rules, determining the classical reduct and then re-associating the certainty degrees with the rules in the reduct. This ensures that there is a 1-on-1 relation between possibilistic answer sets and classical answer sets. Consider the possibilistic normal program

$$\begin{aligned} \mathbf{1} &: lost \leftarrow not\ visible \\ \mathbf{1} &: visible \leftarrow not\ hidden \\ \mathbf{0.5} &: hidden \leftarrow . \end{aligned}$$

This program describes a very simple hiding game. The agent that plays this game will either see the desired item or the item will be hidden (yet we are unsure as to whether the item is actually hidden). When the item is not hidden it is visible and when the item is not visible the agent loses. The unique answer set of the classical program obtained by removing the weights is $\{hidden, lost\}$ and the unique possibilistic answer set, according to the semantics from [1], is $\left\{hidden^{0.5}, lost^{1}\right\}$. While for this example we have a unique possibilistic answer set, in general a possibilistic normal program may have multiple possibilistic answer sets. If we look at the possibilistic answer sets obtained under the semantics from [1], then these are rather restrictive: even though we are unsure as to whether or not the object is hidden, we conclude with absolute certainty that the agent has lost the game. This can also be seen in the program P_{ex} from Section I. The only classical answer set of this program is $\{rain\}$ and the corresponding possibilistic answer set is $\left\{rain^{0.2}\right\}$. The semantics from [1] are thus unable to help us choose being going to the beach or having a barbecue.

The semantics from [2] identify an answer set with a possibility distribution over propositional interpretations. A rule is seen as a constraint on possibility distributions where, intuitively, an answer set corresponds to an epistemic state, in which literals may be more or less certain to hold, and rules are used to reason about which epistemic states are possible, i.e. answer set programming is seen as a form of meta-epistemic reasoning [9]. As in classical ASP, there may be a number of possibility distributions that agree with these constraints (i.e. we may have multiple models). The possibilistic answer sets are then defined as the valuations that correspond with the least specific[1] possibility distribution (i.e. the one that corresponds with the notion of a minimal model in classic ASP). For the normal program mentioned above, the unique possibilistic answer set is $\left\{hidden^{0.5}, visible^{0.5}, lost^{0.5}\right\}$. This conclusion adheres to a different intuition: we are not entirely certain whether the item is hidden, so we are not entirely certain as to whether it is visible nor as to whether the agent has lost the game. If we take a look at the program P_{ex} from Section I, then we do not obtain an answer set. Once again, it does not help us in determining whether we should go to the beach or have a barbecue. The underlying cause of this result can be uncovered if we drop the last rule from P_{ex}, since then we would obtain an infinite family of answer sets of the form $\left\{beach^{c}, bbq^{c'}\right\}$ such that $c + c' = 1$, $c \in [0.4, 0.8]$ and $c' \in [0.2, 0.6]$. Indeed, under the semantics of [2] the idea is that '$not\ a$' is certain to the extent that $\neg a$ is possible. This idea makes sense in this example if we want to evaluate '$not\ rain$'. However, the occurrences of $not\ bbq$ and $not\ beach$ encode that we want to make a choice between two alternatives (provided that it does not rain). This, however, cannot (easily) be expressed under the semantics from [2].

III. Semantics of Uncertain Rules

To arrive at our new semantics for PASP, we start by looking at the intuition of classical ASP. A classical answer set can be seen as an epistemic state, i.e. an answer set determines what an agents knows at a given time. We can reason over which are the possible epistemic states (or answer sets) of an agent using an answer set program. There are several ways in which this view can be refined when certainty scores are attached to rules. Under the semantics from e.g. [2], crisp epistemic states are no longer considered but rather we allow weighted epistemic states, i.e. we let an agent be more or less certain about specific literals.

We can, however, look at these certainties in another way. Rather than considering weighted epistemic states, we can use Boolean epistemic states and use the certainties associated with the rules to express that some epistemic states are more plausible than others. It is clear that considering an invalid

[1]The possibility distribution π that makes minimal commitments. Formally, let π_1 and π_2 be two possibility distributions with the same universe U. We write $\pi_1 \geq \pi_2$ when $\forall u \in U \cdot \pi_1(u) \geq \pi_2(u)$. The least specific possibility distribution in a set, if it exists, is the greatest element w.r.t. $\geq$.

rule may result in incorrect conclusions. Yet, since ASP is non-monotonic, omitting valid information may also allow us to derive additional, erroneous conclusions. Hence, to consider all the possible epistemic states, we need to consider both the answer sets of subprograms of a given program (to find out what should be derived if particular rules of the program were wrong) as well as the answer sets of the program itself, as answer sets of subprograms are not necessarily answer sets of the complete program and vice versa. An answer set is then necessary when it is an answer set of the complete program and when it remains an answer set of all the subprograms from which we have removed uncertain rules. An answer set is possible whenever it is an answer set of the complete program or of a subprogram obtained by removing uncertain rules. Specifically, we can assign a degree of possibility to each subprogram P', which in turn corresponds with an ASP program of which the answer sets may or may not model the epistemic state of the agent in a correct way (as we may have erroneously omitted a valid rule or included an invalid rule). Thus, rather than having a possibility distribution over propositional interpretations we conceptually need a possibility distribution over subprograms $P' \subseteq P$. While there are exponentially many such subprograms, we can encode this possibility distribution by associating a certainty with each rule in our program P.

In particular, a possibilistic rule (r, c) is interpreted as the statement $N(r) \geq c$ where $N(r)$ stands for $N(\{P' \mid P' \subseteq P \text{ and } P' \text{ contains the rule } r\})$, which is analogous to how propositional formulas are interpreted in possibilistic logic [10]. The possibility distribution π_P is then the least specific possibility distribution that satisfies these constraints. A program is considered possible to the extent that it contains all of the certain rules.

Definition 1. Let P be a possibilistic ASP program. We define the possibility distribution π_P over the subsets $P' \subseteq P$. We then have that $\pi_P(P')$ is given by

$$\begin{cases} 1 - \max\{c \mid (\mathbf{c}\colon r) \in P \setminus P'\} & \text{when } P'^* \text{ consistent} \\ 0 & \text{otherwise} \end{cases}$$

It is not hard to see that this corresponds with the least specific possibility distribution that satisfies the constraints, as well as the additional constraint that inconsistent programs are impossible. Furthermore, notice how π_P is a normalized possibility distribution whenever P^* is consistent since then $\pi_P(P) = 1$. Intuitively, this definition states that the less certain the rules are that we omit from the subprogram P', the more possible it is that P' is the correct program.

Now we can define brave and cautious reasoning for PASP programs. Contrary to classical ASP, we not only have answer sets of the complete program P but we also have answer sets for each subprogram $P' \subset P$, for which we also need to take the possibilities into account. Hence we can consider the degree to which a literal is, necessarily or possibly, a brave or cautious consequence of P. This results in a total of four different types of inference.

Definition 2. Let P be a possibilistic ASP program. Let π_P be as in Definition 1. The degree to which it is possible that 'l' is a brave/cautious consequence of P is defined as:

$$\Pi\left(P \models^{\mathrm{b}} l\right) = \max\left\{\pi_P(P') \mid P' \subseteq P \text{ and } P'^* \models^{\mathrm{b}} l\right\}$$
$$\Pi\left(P \models^{\mathrm{c}} l\right) = \max\left\{\pi_P(P') \mid P' \subseteq P \text{ and } P'^* \models^{\mathrm{c}} l\right\}$$

i.e. this is the degree to which some program $P' \subseteq P$ is possible which has 'l' as a brave/cautious consequence. The degree to which it is necessary that P has 'l' as a brave/cautious consequence is defined as:

$$N\left(P \models^{\mathrm{b}} l\right) = 1 - \max\left\{\pi_P(P') \mid P' \subseteq P \text{ and } P'^* \not\models^{\mathrm{b}} l\right\}$$
$$N\left(P \models^{\mathrm{c}} l\right) = 1 - \max\left\{\pi_P(P') \mid P' \subseteq P \text{ and } P'^* \not\models^{\mathrm{c}} l\right\}.$$

Note that this is the degree to which all programs $P' \subseteq P$ that do not have 'l' as a brave/cautious consequence are impossible.

In the following, we will write $P \models^{\mathrm{b}}_{\Pi} l^{\lambda}$ to denote that $\Pi\left(P \models^{\mathrm{b}} l\right) \geq \lambda$, and similar for the notations $P \models^{\mathrm{c}}_{\Pi} l^{\lambda}$, $P \models^{\mathrm{b}}_{N} l^{\lambda}$ and $P \models^{\mathrm{c}}_{N} l^{\lambda}$.

We can now define the possibility degree of an answer set. Each subprogram $P' \subseteq P$ may, by itself, have one or more answer sets. At the same time, two subprograms $P' \subseteq P$ and $P'' \subseteq P$ may have the same answer set, even if $\pi_P(P') \neq \pi_P(P'')$. This leads to the following definition.

Definition 3. Let P be a possibilistic ASP program. Let π_P be the possibility distribution over the subsets $P' \subseteq P$. We define the possibility distribution π_A over the interpretations M:

$$\pi_A(M) = \max\left\{\pi_P(P') \mid M \text{ is an answer set of } P'^*\right\}$$

Note that $\pi_A(M) = 0$ when M is not an answer set of any subprogram $P' \subseteq P$.

Recall that we look at ASP as a form of meta-epistemic reasoning, where one of the answer sets of the program or any of it subprograms corresponds with the actual epistemic state of the agent. We have:

Definition 4. Let π_A be the possibility distribution over the interpretations M. The possibility that l is a literal in the epistemic state of the agent is given by $\Pi(l) = \max\{\pi_A(M) \mid l \in M\}$. The necessity that l is a literal in the epistemic state of the agent is given by $N(l) = 1 - \max\{\pi_A(M) \mid l \notin M\}$.

We then have that $\Pi(l) = \Pi\left(P \models^{\mathrm{b}} l\right)$. Indeed, this readily follows from Definition 2 since $P'^* \models^{\mathrm{b}} l$ is equivalent to stating that there is some answer set M of P'^* with $l \in M$. Similarly we have $N(l) = N\left(P \models^{\mathrm{c}} l\right)$. These two new forms of reasoning thus correspond with two of the forms of reasoning from Definition 2. We now present a number of examples that highlight the proposed semantics.

Example 1. Consider the PASP program P with the rules:

$$\mathbf{0.8}\colon b \leftarrow not\ c \qquad \mathbf{0.3}\colon c \leftarrow d, not\ b \qquad \mathbf{0.9}\colon d \leftarrow .$$

We have that

$$\begin{aligned}
&\pi_P(P) = 1 && \{b, d\}, \{c, d\}\\
&\pi_P(\mathbf{0.8}: b \leftarrow \mathit{not}\ c; \mathbf{0.9}: d \leftarrow) = 0.7 && \{b, d\}\\
&\pi_P(\mathbf{0.3}: c \leftarrow d, \mathit{not}\ b; \mathbf{0.9}: d \leftarrow) = 0.2 && \{c, d\}\\
&\pi_P(\mathbf{0.9}: d \leftarrow) = 0.2 && \{d\}\\
&\pi_P(\mathbf{0.8}: b \leftarrow \mathit{not}\ c; \mathbf{0.3}: c \leftarrow d, \mathit{not}\ b) = 0.1 && \{b\}\\
&\pi_P(\mathbf{0.8}: b \leftarrow \mathit{not}\ c) = 0.1 && \{b\}\\
&\pi_P(\mathbf{0.3}: c \leftarrow d, \mathit{not}\ b) = 0.1 && \{\}\\
&\pi_P(\{\}) = 0.1 && \{\}.
\end{aligned}$$

where the possibility associated with the subprogram is shown on the left and the unique classical answer set of the subprogram is shown on the right. We can verify that we obtain the following conclusions:

$$\begin{aligned}
&P \models^{\mathrm{b}}_N \left\{b^{0.8}, c^{0.3}, d^{0.9}\right\} && P \models^{\mathrm{c}}_N \left\{b^{0}, c^{0}, d^{0.9}\right\}\\
&P \models^{\mathrm{b}}_\Pi \left\{b^{1}, c^{1}, d^{1}\right\} && P \models^{\mathrm{c}}_\Pi \left\{b^{0.7}, c^{0.2}, d^{1}\right\}.
\end{aligned}$$

It readily follows that $\Pi(b) = 1, \Pi(c) = 1$ and $\Pi(d) = 1$ and $N(b) = 0, N(c) = 0$ and $N(d) = 0.9$. The different semantics for PASP do not necessarily agree with each other, as illustrated in the next example.

Example 2. Consider the PASP program P from Section II with the rules:

$$\begin{aligned}
\mathbf{1}&: \mathit{lost} \leftarrow \mathit{not}\ \mathit{visible}\\
\mathbf{1}&: \mathit{visible} \leftarrow \mathit{not}\ \mathit{hidden}\\
\mathbf{0.5}&: \mathit{hidden} \leftarrow
\end{aligned}$$

For brevity, we name these rules from top to bottom r_1, r_2 and r_3. We have that

$$\begin{aligned}
&\pi_P(P) = 1 && \{\mathit{hidden}, \mathit{lost}\}\\
&\pi_P(r_1, r_2) = 0.5 && \{\mathit{visible}\}
\end{aligned}$$

whereas the possibility of all the other subprograms $P' \subseteq P$ is 0. Since each subprogram has a unique answer set, we have that brave and cautious reasoning coincide. The cautious conclusions that we obtain are:

$$\begin{aligned}
&P \models^{\mathrm{c}}_N \left\{\mathit{hidden}^{0.5}, \mathit{visible}^{0}, \mathit{lost}^{0.5}\right\}\\
&P \models^{\mathrm{c}}_\Pi \left\{\mathit{hidden}^{1}, \mathit{visible}^{0.5}, \mathit{lost}^{1}\right\}.
\end{aligned}$$

These conclusions do not agree with either the semantics from [1] or [2], yet provide an intuitively satisfiable answer to the outcome of the game that the agent plays. Indeed, we can conclude that it is entirely possible that the agent has lost (since $P \models^{\mathrm{c}}_\Pi \mathit{lost}^{1}$), while at the same time we know that this is not necessarily so (since $P \models^{\mathrm{c}}_N \mathit{lost}^{0.5}$).

It is furthermore easy to verify that the new semantics are a proper extension of classical ASP, i.e. when we assume that each rule is totally certain we recover the classical answer sets.

Example 3. Consider the PASP program P with the set of rules $\{\mathbf{1}: a \leftarrow \mathit{not}\ b, \mathbf{1}: b \leftarrow \mathit{not}\ a\}$. We have that

$$\pi_P(P) = 1 \qquad \{a\}, \{b\}$$

whereas the possibility of all the others subprogram $P' \subseteq P$ is 0. In this example we cannot derive any literals with a degree different from 0 under cautious reasoning, while we do find $P \models^{\mathrm{b}}_N \left\{a^1, b^1\right\}$ and $P \models^{\mathrm{b}}_\Pi \left\{a^1, b^1\right\}$. This corresponds exactly with the brave/cautious conclusions that can be derived from the classical program $P = \{a \leftarrow \mathit{not}\ b; b \leftarrow \mathit{not}\ a\}$. This result holds trivially for all PASP programs that only use certainties $c = 1$. Indeed, all the subsets of P have a possibility of 0, hence the only possible subset of P is P itself.

The new semantics overcome some of the problems inherent in the semantics of [1] and adhere to a different intuition than the semantics of [2]. One of the problems with [1] is that the semantics do not behave intuitively when conclusions are made through NAF. This is because the semantics rely on the classical reduct and do not take the certainty of the literals into account when dealing with NAF. For example:

$$\mathbf{0.1}: \mathit{canceled} \leftarrow \qquad \mathbf{1}: \mathit{concert} \leftarrow \mathit{not}\ \mathit{canceled}.$$

This program intuitively states that we have a low certainty that a concert is canceled and that the concert will only take place if there is no indication that it is canceled. The semantics from [1] give rise to the answer set $\left\{\mathit{canceled}^{0.1}\right\}$, i.e. they only state that the concert is canceled. The semantics presented in this paper conclude $\left\{\mathit{canceled}^{0.1}, \mathit{concert}^{0.9}\right\}$, i.e. the concert still takes place, though we are less certain of our conclusion. This corresponds with the semantics from [2]. Nevertheless, the semantics from [2] follow a different intuition. For example, the program with the rules

$$\mathbf{c}: \mathit{left} \leftarrow \mathit{not}\ \mathit{right} \quad \mathbf{c'}: \mathit{right} \leftarrow \mathit{not}\ \mathit{left} \quad \mathbf{1}: \leftarrow \mathit{left}, \mathit{right}$$

with $c, c' \in]0, 1[$ has no answer sets. Nevertheless, intuitively, the program merely states that we need to choose either left or right, i.e. an exclusive choice. This behavior can also be seen in P_{ex} from Section I where we need to choose either beach or bbq, yet are unable to. The semantics from this paper are able to make such an exclusive choice, as in the next example.

Example 4. Consider P_{ex} from Section I. Under the semantics proposed in this paper, we obtain:

$$\begin{aligned}
P_{\text{ex}} &\models^{\mathrm{b}}_\Pi \left\{\mathit{rain}^{1}, \mathit{beach}^{0.8}, \mathit{bbq}^{0.8}\right\}\\
P_{\text{ex}} &\models^{\mathrm{b}}_N \left\{\mathit{rain}^{1}, \mathit{beach}^{0.4}, \mathit{bbq}^{0.2}\right\}\\
P_{\text{ex}} &\models^{\mathrm{c}}_\Pi \left\{\mathit{rain}^{0.2}, \mathit{beach}^{0}, \mathit{bbq}^{0}\right\}\\
P_{\text{ex}} &\models^{\mathrm{c}}_N \left\{\mathit{rain}^{0.2}, \mathit{beach}^{0}, \mathit{bbq}^{0}\right\}
\end{aligned}$$

Since the necessity associated with beach is always higher than, or equal to, the necessity associated with bbq for all reasoning tasks, we can conclude that we are more likely to go to the beach than to have a barbecue.

We now give a more elaborate example to illustrate that the semantics presented in this paper do not have the aforementioned issues, even if both are present at the same time.

Example 5. Triage at an accident site with a large number of casualties is an essential part of medical treatment when resources are limited. With the help of triage, it becomes possible to distinguish which casualties can wait for medical

attention at a hospital and which casualties need to be treated on the spot. For brevity of this example, we consider a triage system with three levels. The casualty may have minor injuries ($minor$), which means that the person can wait for treatment at the hospital. The casualty may need to be treated immediately because of life-threatening, yet treatable injuries ($nowait$). The final category is beyond urgency ($beyond$) and encompasses those casualties which are so severely injured that, for the time being, medical attention is better directed towards casualties in the $nowait$ category as the chances of survival of casualties in this latter category are far higher.

A rescue helper is faced with a casualty with extensive external injuries ($extensive$), which indicates that he/she either falls in the $nowait$ or $beyond$ category. The casualty is faintly moaning ($moaning$), which, with a very low certainty, is an indication of the casualty still being conscious ($conscious$). Similarly, the casualty is exhibiting a bleeding nose ($nosebleed$), which might indicate internal bleeding ($internal$). The rescue helper would be a lot more certain that the casualty is experiencing internal bleeding when he/she also had low blood pressure ($lowblood$), but this has not been established. Whenever the casualty does not appear to be conscious, he/she is assumed to be in the $nowait$ or $beyond$ category. When there is no indication of internal bleeding, the casualty is in the $nowait$ category. A classification in one of the categories is never entirely certain since it is not possible, due to time constraints, to perform all the required tests. We have the program with the rules:

$$\begin{aligned}
&\mathbf{1}: extensive \leftarrow \\
&\mathbf{0.9}: minor \leftarrow not\ extensive \\
&\mathbf{1}: moaning \leftarrow \\
&\mathbf{0.1}: conscious \leftarrow moaning \\
&\mathbf{0.9}: nowait \leftarrow not\ beyond, not\ internal, \\
&\quad not\ conscious, extensive \\
&\mathbf{0.9}: beyond \leftarrow not\ nowait, not\ conscious, extensive \\
&\mathbf{1}: nosebleed \leftarrow \\
&\mathbf{0.1}: internal \leftarrow nosebleed \\
&\mathbf{0.7}: internal \leftarrow nosebleed, lowblood \\
&\mathbf{1}: \leftarrow nowait, beyond, extensive \\
&\mathbf{1}: \leftarrow not\ nowait, not\ beyond, extensive
\end{aligned}$$

The last two rules encode that one and exactly one of the categories needs to be chosen when the casualty has extensive injuries, a requirement for an efficient triage. Notice that at least one rule needs to be omitted to make the program consistent. Under the semantics of [1] and the semantics [2] we obtain no answer sets. Indeed, the semantics of [1] are unable to take the low certainty of the literal $conscious$ into account when reasoning with NAF. The semantics of [2], on the other hand, are unable to choose between $nowait$ and $beyond$. Indeed, we will obtain an infinite number of answer sets such that the sum of the necessities of $nowait$ and $beyond$ equals 0.9 and such that the necessity for both $nowait$ and $beyond$ will be higher or equal to 0.1. Under the semantics proposed in this paper, however, we obtain that $P \models^{\mathrm{b}}_{\Pi} \{beyond^{0.9}, nowait^{0.9}\}$, $P \models^{\mathrm{c}}_{\Pi} \{beyond^{0.9}, nowait^{0.1}\}$, $P \models^{\mathrm{b}}_{N} \{beyond^{0.9}, nowait^{0.1}\}$ and $P \models^{\mathrm{c}}_{N} \{beyond^{0.1}, nowait^{0.1}\}$. The new semantics are thus capable of arriving at the desired conclusion. Indeed, since the necessity associated with $beyond$ is higher or equal to the necessity associated with $nowait$ for all reasoning tasks, a reasonable classification for the casualty is $beyond$. This corresponds with our intuition, as a number of indications hint towards this worst case scenario (e.g. the bleeding nose). If we added the fact that the casualty has low blood pressure, then even a brave conclusion with possibility measures would indicate that the casualty is beyond urgency, i.e. it would further reaffirm our conclusion.

IV. Complexity Results

We now analyze the complexity of the decision problems associated with the inference types from Definition 2. More specifically, we are interested in the complexity of determining whether for some PASP program P and some literal l we can bravely/cautiously derive that l is a conclusion of P with a necessity/possibility degree of at least λ. Recall that the complexity class Σ^{P}_2, sometimes also written $\mathsf{NP}^{\mathsf{NP}}$, is the class of problems that can be solved in polynomial time on a non-deterministic machine with an NP oracle, i.e. assuming access to a procedure that can solve NP problems in constant time [11]. The complexity class Π^{P}_2 contains those problems whose complement is in Σ^{P}_2, i.e. $\Pi^{\mathsf{P}}_2 = \mathsf{co}\Sigma^{\mathsf{P}}_2$ [11]. Deciding the validity of a Quantified Boolean Formula (QBF) $\phi = \exists X_1 \forall X_2 \cdot p(X_1, X_2)$ with $p(X_1, X_2)$ a formula in disjunctive normal form (DNF) over the set of variables $X_1 \cup X_2$ is the canonical Σ^{P}_2-complete problem. Deciding the validity of a QBF $\phi = \forall X_1 \exists X_2 \cdot p(X_1, X_2)$ with $p(X_1, X_2)$ in conjunctive normal form (CNF) is the canonical Π^{P}_2-complete problem. Brave (resp. cautious) reasoning for normal programs is known to be NP-complete (resp. coNP-complete) [6].

Proposition 1. *Let P be a possibilistic normal program. Deciding whether $N(P \models^{\mathrm{c}} l) \geq \lambda$ is in coNP.*

Proof: We will show that the complementary problem is in NP. To determine whether $N(P \models^{\mathrm{c}} l) \not\geq \lambda$ we guess on the one hand a subset P' of rules from P such that $\pi_P(P') > 1-\lambda$ and on the other hand an interpretation M, which does not include 'a'. Given such a non-deterministic guess, we can verify in polynomial time that M is an answer set of P' and hence that $P' \not\models^{\mathrm{c}} a$. From Definition 2 we know that $N(P \models^{\mathrm{c}} l) = 1 - \max\{\pi_P(Q) \mid Q \subseteq P \text{ and } Q^* \not\models^{\mathrm{c}} a\} \leq 1 - \pi_P(P') < \lambda$. In other words: determining whether $N(P \models^{\mathrm{c}} l) \not\geq \lambda$ is in NP. Deciding whether $N(P \models^{\mathrm{c}} l) \geq \lambda$ is thus in coNP. ■

Proposition 2. *Let P be a possibilistic normal program. Deciding whether $N\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is in Π^{P}_2.*

Proof: We will show that the complementary problem is in Σ^{P}_2. To determine whether $N\left(P \models^{\mathrm{b}} l\right) \not\geq \lambda$ we guess a subset P' of rules from P such that $\pi_P(P') > 1 - \lambda$. Notice that, contrary to Proposition 1, we cannot simply guess an interpretation to verify that $N\left(P \models^{\mathrm{b}} l\right) \not\geq \lambda$. Indeed, determining

whether $P' \not\models^{\mathrm{b}} a$ requires that there is not a single answer set in which 'a' is not contained, whereas for determining whether $P' \not\models^{\mathrm{c}} a$ it suffices to find exactly one answer set in which 'a' is not contained. Therefore, given a non-deterministic guess for P', we rely on an NP-oracle [6] to verify in constant time that $P' \not\models^{\mathrm{b}} a$. Similar as in Proposition 1 this gives us a counterexample for $N\left(P \models^{\mathrm{b}} l\right) \geq \lambda$. Hence determining whether $N\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is in $\mathsf{co}\left(\mathsf{NP}^{\mathsf{NP}}\right)$, i.e. in Π_2^{P}. ■

Proposition 3. *Let P be a possibilistic normal program. Deciding whether $\Pi\left(P \models^{\mathrm{c}} l\right) \geq \lambda$ is in Σ_2^{P}.*

Proposition 4. *Let P be a possibilistic normal program. Deciding whether $\Pi\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is in NP.*

The proofs of the last two propositions are entirely analogous to the proofs of Proposition 1 and Proposition 2. Moreover, the following hardness results are trivially carried over from the hardness of brave and cautious reasoning in classical ASP [6]:

Proposition 5. *Let P be a possibilistic normal program. Deciding whether $N\left(P \models^{\mathrm{c}} l\right) \geq \lambda$ is coNP-hard ($\lambda > 0$).*

Proposition 6. *Let P be a possibilistic normal program. Deciding whether $\Pi\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is NP-hard ($\lambda > 0$).*

The next two results are more involved and require a simulation of QBFs.

Definition 5. Let $\phi = \exists X_1 \forall X_2 \cdot p(X_1, X_2)$ be a QBF such that $p(X_1, X_2) = \theta_1 \vee ... \vee \theta_n$ a formula in DNF with X_i sets of variables. We define the possibilistic normal program P_ϕ corresponding to ϕ as

$$P_\phi = \{\mathbf{0.5} : x \leftarrow \mid x \in X_1\} \cup \{\mathbf{0.5} : \neg x \leftarrow \mid x \in X_1\} \quad (1)$$
$$\cup \{(\mathbf{1} : x \leftarrow not\ \neg x), (\mathbf{1} : \neg x \leftarrow not\ x) \mid x \in X_2\} \quad (2)$$
$$\cup \{\mathbf{1} : sat \leftarrow \theta_t \mid 1 \leq t \leq n\} \quad (3)$$

where we identify the conjunction of literals θ_t in (3) with a set of literals.

Example 6. Consider the QBF $\phi = \exists p_1, p_2 \forall q \cdot (p_1 \wedge q) \vee (p_2 \wedge \neg q)$. The possibilistic normal program P_ϕ is then

$$\mathbf{0.5} : p_1 \leftarrow \qquad \mathbf{0.5} : \neg p_1 \leftarrow$$
$$\mathbf{0.5} : p_2 \leftarrow \qquad \mathbf{0.5} : \neg p_2 \leftarrow$$
$$\mathbf{1} : q \leftarrow not\ \neg q \qquad \mathbf{1} : \neg q \leftarrow not\ q$$
$$\mathbf{1} : sat \leftarrow p_1, q \qquad \mathbf{1} : sat \leftarrow p_2, \neg q$$

Proposition 7. *Let P be a possibilistic normal program. Deciding whether $\Pi\left(P \models^{\mathrm{c}} l\right) \geq \lambda$ is Σ_2^{P}-hard ($\lambda > 0$).*

Proof: We reduce the problem of determining the satisfiability of a QBF of the form $\phi = \exists X_1 \forall X_2 \cdot p(X_1, X_2)$ with $p(X_1, X_2)$ in DNF to the problem of deciding whether $\Pi\left(P \models^{\mathrm{c}} l\right) \geq \lambda$. Specifically, we use the possibilistic normal program P_ϕ that simulates ϕ from Definition 5 and show that the QBF is satisfiable if and only if $\Pi\left(P_\phi \models^{\mathrm{c}} sat\right) \geq 0.5$.

The rules in (1) ensure that there are at least as many subprograms $P' \subseteq P_\phi$ as there are interpretations of X_1. The subprograms P' with $\pi_{P_\phi}(P') > 0$ then contain the rules (2) that generate as many answer sets as there are interpretations of X_2. The rule (3) ensures that 'sat' is contained in the classical answer set whenever for a chosen interpretation of X_1 and X_2 it holds that $p(X_1, X_2)$ is satisfied. Notice that the certainty attached to the rules ensures that removing any of the rules from (2) or (3) results in $\pi_{P_\phi}(P') = 0$, i.e. it indicates that these rules are completely necessary. We then have that $\Pi\left(P_\phi \models^{\mathrm{c}} sat\right) \geq 0.5$ if and only if the QBF is satisfiable. Indeed, from the construction of P_ϕ we know that every consistent interpretation of X_1 will only have a subset of the rules from (1) and, more specifically, there will be a one-on-one relation between the consistent interpretation of X_1 and the subprograms with a possibility of 0.5. Also, $P'^* \models^{\mathrm{c}} sat$ if and only if P' corresponds to an interpretation of X_1 such that $p(X_1, X_2)$ is satisfied for every interpretation of X_2. Using the possibility measure (i.e. finding $\max\left\{\pi_{P_\phi}(P') \mid P' \subseteq P_\phi \text{ and } P'^* \models^{\mathrm{c}} sat\right\}$) implies that the QBF is satisfied whenever we find at least one such an interpretation X_1.

Some of the subprograms of P_ϕ may correspond to partial or inconsistent interpretations of X_1. However, the subprograms P' corresponding with inconsistent interpretations have $\pi_{P_\phi}(P') = 0$ by definition and can therefore never be used to derive $\Pi\left(P_\phi \models^{\mathrm{c}} sat\right) \geq 0.5$. Furthermore, any subprogram P' with incomplete assignments for the variables in X_1 from which we can conclude that $P'^* \models^{\mathrm{c}} sat$ can trivially be extended to a subprogram P'' to which we add some rules from (1) to complete the assignment for the variables in X_1 and we will still be able to conclude that $P''^* \models^{\mathrm{c}} sat$. Thus, these additional subprograms do not affect our ability to derive $\Pi\left(P_\phi \models^{\mathrm{c}} sat\right) \geq 0.5$. ■

Proposition 8. *Let P be a possibilistic normal program. Deciding whether $N\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is Π_2^{P}-hard ($\lambda > 0$).*

Proof: Let Q be the program defined as $P \cup \{\mathbf{1} : x \leftarrow not\ l\}$ with x a fresh literal. Then $N\left(Q^* \models^{\mathrm{b}} l\right) \geq \lambda$ if and only if $\Pi\left(Q^* \models^{\mathrm{c}} x\right) \leq 1 - \lambda$. Indeed, we know from Definition 2 that $N\left(Q^* \models^{\mathrm{b}} l\right) \geq \lambda$ whenever $1 - \max\left\{\pi_P(P') \mid P'^* \not\models^{\mathrm{b}} l\right\} \geq \lambda$, i.e. whenever $\max\left\{\pi_Q(P') \mid P' \subseteq Q \text{ and } P'^* \not\models^{\mathrm{b}} l\right\} \leq 1 - \lambda$. Because the newly added rule $x \leftarrow not\ l$ will be in every subprogram P' with $\pi_Q(P') > 0$ (since the certainty attached to this rule is 1), we know that there is at least one answer set in which l is true if and only if it is not the case that x is true in every answer set. Thus, the previous inequality is equivalent to $\max\left\{\pi_Q(P') \mid P' \subseteq Q \text{ and } P'^* \models^{\mathrm{c}} x\right\} \leq 1 - \lambda$ and, by applying Definition 2, to $\Pi\left(Q^* \models^{\mathrm{c}} x\right) \leq 1 - \lambda$. Since the set of certainty values associated with the rules is finite, this equation is equivalent to $\Pi\left(Q^* \models^{\mathrm{c}} x\right) < \lambda'$ for some λ'. Hence $\neg(\Pi\left(Q^* \models^{\mathrm{c}} x\right) \geq \lambda')$. This problem is therefore the complement of the decision problem from Proposition 7. Thus deciding whether $N\left(P \models^{\mathrm{b}} l\right) \geq \lambda$ is Π_2^{P}-hard. ■

V. Related Work

Combining logic and uncertainty in a single framework is an active topic of research. For example, the work from [12]

on probabilistic logic is one of the first generalizations of propositional logic to a logic that allows to associate probabilistic values to formulas. The semantics are defined in terms of probability distributions over possible worlds, where the probability attached to a formula corresponds with the probability that the real world is among those possible worlds that make the formula true. These ideas have later been extended to probabilistic logic programming [13]. Similar work had already been done in [14], which is one of the earliest works to combine probability theory with non-monotononic negation in the setting of probabilistic deductive databases.

Another popular approach for dealing with uncertainty are Markov Logic Networks [15] which use Markov networks [16] to compactly describe a probability distribution over possible worlds. Other approaches include, for example, the work in Bayesian Logic Programming [17], where a generalization of Bayesian networks is used to reason over Horn clauses. Specifically within the domain of ASP [5], we find the work from [18], which combines ASP with probability theory [19]. Probabilistic atoms are used to encode uncertain information which, intuitively, describes the probability that an associated random variable will take on a given value.

Aside from probability theory, possibility theory [7] has also been used by many authors for dealing with uncertainty in a non-monotonic setting. The work in [20] combines defeasible logic, a form of non-monotonic reasoning involving both strict and defeasible rules, with possibility theory in a single framework. This allows, among other things, to resolve conflicts between contradictory goals. Possibility theory has also been combined with argumentation frameworks, e.g. in [21], where revision rules allow an agent to revise its beliefs and goals.

One of the earliest works that combines possibility theory with stable models is [22], where a compositional version of possibilistic logic is used as a logic of graded truth to deal with uncertainty. More recently, [1] combined possibility theory with ASP, resulting in the PASP framework. Recently, other semantics for PASP have been envisaged, including [2] (as discussed in Section II) and the work on pstable models [23], [24], which is a framework based on the fusion of ASP and paraconsistent logic. Such pstable models are closer to possibilistic logic [10] and to the intuition of classical models, rather than to the intuition of stable models as in our approach.

VI. Conclusions

We have introduced new semantics for PASP based on the idea that a PASP program P is an incomplete specification of an ASP program. Every classical ASP program can be seen as an encoding of the possible epistemic states of an agent. The idea underlying the PASP semantics then boils down to keeping the idea of answer sets as (Boolean) epistemic states, but use the certainty degrees attached to the rules to express that some epistemic states are more plausible than others. As such we are able to construct a possibility distribution over the universe of subsets of P. This gives rise to four distinct types of inference. We have examined the complexity of these inference types and showed that two of these types of inference are no more complex than in classical ASP, whereas the others are located one level higher in the polynomial hierarchy.

In future work, we want to extend our approach beyond starting from the certainty of individual rules. Rather, we want to start from a causal network, where we graphically model the relations between rules, to be able to more precisely describe the uncertainty that the agent faces. Possibilistic causal networks [25] are the possibilistic equivalent of Bayesian probabilistic networks [26] and are an obvious candidate.

References

[1] P. Nicolas, L. Garcia, I. Stéphan, and C. Lefèvre, "Possibilistic uncertainty handling for answer set programming," *Annals of Mathematics and Artificial Intelligence*, vol. 47, no. 1–2, pp. 139–181, 2006.

[2] K. Bauters, S. Schockaert, M. De Cock, and D. Vermeir, "Possibilistic answer set programming revisited," in *Proc. of UAI'10*, 2010.

[3] C. V. Damásio and L. M. Pereira, "Monotonic and residuated logic programs," in *Proc. of ECSQARU'01*, 2001, pp. 748–759.

[4] K. Bauters, S. Schockaert, M. De Cock, and D. Vermeir, "Weak and strong disjunction in possibilistic ASP," in *Proc. of SUM'11*, 2011.

[5] M. Gelfond and V. Lifzchitz, "The stable model semantics for logic programming," in *Proc. of ICLP'98*, 1988, pp. 1081–1086.

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

[7] D. Dubois and H. Prade, *Possibility theory: an approach to computerized processing of uncertainty*. Plenum Press, 1988.

[8] G. L. S. Shackle, *Decision Order & Time in Human Affairs*. Cambridge University Press, 1961.

[9] D. Dubois, H. Prade, and S. Schockaert, "Stable models in generalized possibilistic logic," *Proc. of KR'12*, 2012, to appear.

[10] D. Dubois, J. Lang, and H. Prade, "Possibilistic logic," *Handbook of Logic for Artificial Intelligence and Logic Programming*, vol. 3, no. 1, pp. 439–513, 1994.

[11] C. Papadimitriou, *Computational complexity*. Addison-Wesley, 1994.

[12] N. J. Nilsson, "Probabilistic logic," *Artificial Intelligence*, vol. 28, no. 1, pp. 71–87, 1986.

[13] T. Lukasiewicz, "Probabilistic default reasoning with conditional constraints," *Annals of Mathematics and Artificial Intelligence*, vol. 34, no. 1–3, pp. 35–88, 2002.

[14] R. Ng and V. Subrahmanian, "Stable model semantics for probabilistic deductive databases," in *Proc. of ISMIS'91*, vol. 542, 1991, pp. 162–171.

[15] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[16] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[17] K. Kersting and L. D. Raedt, "Bayesian logic programs," *CoRR*, 2001.

[18] C. Baral, M. Gelfond, and N. Rushton, "Probabilistic reasoning with answer sets," *Theory and Practice of Logic Programming*, vol. 9, no. 1, pp. 57–144, 2009.

[19] E. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[20] C. Chesñevar, G. Simari, T. Alsinet, and L. Godo, "A logic programming framework for possibilistic argumentation with vague knowledge," in *Proc. of UAI'04*, 2004.

[21] L. Amgoud and H. Prade, "Reaching agreement through argumentation: A possibilistic approach," in *Proc. of KR'04*, 2004, pp. 175–182.

[22] G. Wagner, "Negation in fuzzy and possibilistic logic programs," in *Proc. of LPSC'98*, 1998, pp. 113–128.

[23] R. Confalonieri, J. C. Nieves, and J. Vázquez-Salceda, "Pstable semantics for logic programs with possibilistic ordered disjunction," in *Proc. of AI*IA'09*, 2009, pp. 52–61.

[24] M. Osorio, J. A. N. Pérez, J. R. A. Ramírez, and V. B. Macías, "Logics with common weak completions," *Journal of Logic and Computation*, vol. 16, no. 6, pp. 867–890, 2006.

[25] S. Benferhat and S. Smaoui, "Possibilistic causal networks for handling interventions: A new propagation algorithm," in *Proc. of AAAI'07*, 2007.

[26] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.