Towards a New Semantics for Possibilistic Answer Sets

Julien Hué, Matthias Westphal, and Stefan Wölfl

Albert-Ludwigs-Universität Freiburg, Department of Computer Science, Germany
{hue,westpham,woelfl}@informatik.uni-freiburg.de

Abstract. Possibilistic Answer Set Programming is an extension of the standard ASP framework that allows for attaching degrees of certainty to the rules in ASP programs. In the literature, several semantics for such PASP-programs have been presented, each of them having particular strengths and weaknesses. In this work we present a new semantics that employs so-called iota-answer sets, a solution concept introduced by Gebser et al. (2009), in order to find solutions for standard ASP programs with odd cycles or auto-blocking rules. This is achieved by considering maximal subsets of a given ASP program for which answer sets exist. The main idea of our work is to integrate iota-semantics into the possibilistic framework in such a way that degrees of certainty are not only assigned to atoms mentioned in the answer sets, but also to the answer sets themselves. Our approach gives more satisfactory solutions and avoids counter-intuitive examples arising in the other approaches. We compare our approach to existing ones and present a translation into the standard ASP framework allowing the computation of solutions by existing tools.

1 Introduction

Answer Set Programming (ASP) [1] is a logic programming formalism which is nowadays one of the main paradigms for nonmonotonic reasoning. In view of application contexts, however, the framework has some limitations. Quite often, when a knowledge base is to be set up, the modeler is not only aware of the nonmonotonic nature of some general rules to be represented, but also considers some of these rules more plausible than others. Within the standard ASP framework it is not possible to order rules according to their certainty or plausibility in an explicit manner. Of course, the modeler can try to modify the knowledge base by introspecting which rules are applied under which conditions in order to obtain some expected solution to a reasoning task on the knowledge base (see, e.g., [2]). But if the problem instance has several solution candidates it is still not possible to rank them, i.e., to estimate which of them is (or should be considered) more plausible than others.

One way to remedy this deficiency is to consider knowledge bases equipped with probability distributions. In this context different strands of research can be found in the literature. For the more general context of equipping conditionals with probabilities we refer to [3], and [4]. But also in the context of logic and answer set programming, probabilistic approaches have been discussed (see., e.g., Poole [5], Lukasiewicz [6], Baral et al. [7]). The problem with such approaches is that most of the time exact probability values (and even lower or upper bounds of them) are not available or hard to argue for. A typical situation is that the modeler has just some qualitative ranking between the rules in the knowledge base in mind, and (maybe even more importantly) that she is also only interested in an ordinal-scale ranking of the possible solutions of the given problem.

In this situation, ideas from possibilistic logic [8] seem promising to deal with plausibility degrees of general rules. Possibilistic logic is a framework for representing uncertainty (in a propositional logic context) by using a pair of possibility and necessity measures. These measures are understood qualitatively and are used for rank-ordering interpretations. To combine the ideas of possibilistic logic and answer set programming (referred to as *possibilistic answer set programming*) several approaches have been proposed in the literature, for example, by Bauters et al. [9], and Nicolas et al. [10]. Each of these approaches has particular strengths and weaknesses, which will be discussed later in more detail. The reader might also consult [11] which presents a variation of stable semantics called pstable semantics associated with possibilistic logic.

In this paper, we present a new semantics that is based on so-called ι-answer sets. The ι-answer set semantics has been introduced by Gebser et al. [12] in order to solve problems arising in standard logic programs, e.g., with odd cycles or with auto-blocking rules. The main idea is to consider subsets of the program instead of the complete program to check whether it is possible to entail a set of atoms. The semantics we propose for possibilistic answer sets follows the ideas of ι-answer set semantics. As we will see, the proposed semantics naturally leads to not only associating a necessity value to each atom within a stable model, but also a possibility value to the stable models themselves. We argue that the new semantics, while still being NP-complete, is closer to the philosophy underlying both possibilistic logic and ASP than other semantics in the literature.

Structure of the paper. We start by presenting in the next section the necessary background and notations, and discuss in more detail the mentioned other semantic approaches for possibilistic logic programs. We then present our new semantics, and discuss its formal properties. Finally, we present a translation of possibilistic answer set programs into classic answer set programs which allows to use existing tools for computing solutions.

2 Background and Notations

2.1 Answer Set Programming

A logic program (or *normal logic program*) is a (finite) set of rules of the form

$$r \; : \; a \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n.$$

where a, b_i and c_j are propositional atoms. The keyword *not* denotes negation as failure. The atom a is called the head of the rule (denoted by $\mathit{head}(r)$) and $b_1, \ldots, b_m$, not $c_1, \ldots,$ not c_n is called the body of the rule. The set of all atoms b_i and c_j that make up the body of the rule is denoted by $\mathit{body}(r)$. The body can be divided into a positive and negative part. Atoms b_i represent the positive body, denoted by $\mathit{body}^+(r)$, and atoms

c_j represent the negative body, denoted by $body^-(r)$. Thus, $body(r) = body^+(r) \cup body^-(r)$ and we sometimes write r as $head(r) \leftarrow body^+(r),$ not $body^-(r)$. Further, we extend this notation to logic programs P in the obvious way: $body^+(P) = \{a \mid a \in body^+(r)$ for some $r \in P\}$ and $body^-(P) = \{a \mid a \in body^-(r)$ for some $r \in P\}$. We denote by $atoms(P)$ the set of atoms that occur in P.

Intuitively, a rule can be understood as follows: if each of the atoms b_i of the positive body of the rule is true and none of the atoms c_j of the negative body is true, then the head of the rule can be inferred. Given rule r, we denote by r^+ the rule $head(r) \leftarrow body^+(r)$. A rule r with empty head is called an *integrity constraint* (we often write such rules in the form $\bot \leftarrow \ldots$). Similarly, the body of a rule can be empty, thus stating that the head is a *fact*.

Atoms a as well as their default negation not a are called *literals*. A *basic program* is a logic program where all rules are of the form $r : a \leftarrow b_1, \ldots, b_m$, i.e., with an empty negative body and thus without negation as failure.

Definition 1. *A set of atoms A is said to be* closed under a basic program P *if for each rule $r \in P$, $head(r) \in A$ whenever $body(r) \subseteq A$. The smallest set closed under a basic program P is denoted by $CN(P)$, and referred to as the set of* consequences *of the program. For arbitrary programs we write $CN^+(P)$ for the set $CN(\{r^+ \mid r \in P\})$.*

A *stable model* of a program is a set of atoms that represents a set of consequences consistent with the beliefs expressed by the program, but also has the property that the presence of each of the atoms in the set is justified. In other words, a stable model is necessarily minimal with respect to set inclusion, that is, a proper subset of a stable model cannot be a stable model. More formally, stable models can be defined in terms of the so-called Gelfond-Lifschitz reduct [13]:

Definition 2. *The* Gelfond-Lifschitz reduct *of a program P by a set of atoms A is defined as the following set of rules: $P^A = \left\{r^+ \mid r \in P \text{ and } body^-(r) \cap A = \emptyset\right\}$. Given a logic program P, a set of atoms A is called a* stable model of P *if and only if $CN(P^A) = A$.*

In the literature, the terms answer set and stable model are mostly used in an equivalent way. For a more comprehensive introduction to answer set semantics we refer to [14,1].

2.2 ι-Answer Sets

Gebser et al. [12] introduce the notion of ι-answer sets. This concept allows for incrementally constructing solutions to a given normal logic program. Contrary to standard answer set semantics, the ι-answer set semantics has the advantage that we can always talk about solutions of a program, even when stable models do not exist.

Definition 3. *Let P be a logic program and A be a set of atoms. Then A is called an* ι-answer set *of P if $A = CN^+(Q)$ for some $\subseteq$-maximal $Q \subseteq P$ such that (i) $body^+(Q) \subseteq CN^+(Q)$ and (ii) $body^-(Q) \cap CN^+(Q) = \emptyset$.*

It is also possible to characterize the ι-answer sets in terms of applied rules.

Definition 4. *Let P be a logic program and A be a set of atoms. Then the set of applied rules of P for A is defined as:*

$$\mathit{App}_P(A) = \big\{ r \in P \mid \mathit{body}^+(r) \subseteq A, \mathit{body}^-(r) \cap A = \emptyset, \mathit{head}(r) \in A \big\}.$$

It can be shown that a set of atoms A is an ι-answer set of P if and only if $A = \mathit{CN}^+(\mathit{App}_P(A))$ and if for each $r \in P \setminus \mathit{App}_P(A)$, at least one of the following conditions holds true: (i) $\mathit{body}^+(r) \not\subseteq A$, (ii) $\mathit{body}^-(r) \cap A \neq \emptyset$, or (iii) $\mathit{head}(r) \in \mathit{body}^-(\mathit{App}_P(A) \cup \{r\})$.

2.3 Possibilistic Logic

We consider a finite propositional language $\mathcal{L}$. The set of all interpretations over $\mathcal{L}$ is denoted by Ω. Possibilistic logic [8] is defined in terms of a possibility distribution $\pi\colon \Omega \to [0,1]$ representing how plausible an interpretation is with regard to the available knowledge. For an interpretation $\omega \in \Omega$, $\pi(\omega) = 0$ means that ω is considered impossible and $\pi(\omega) = 1$ means that there is no contradiction with assuming ω to be true. The possibility distribution only represents a *preordering* over the interpretations, i.e., $\pi(\omega) > \pi(\omega')$ expresses that ω is considered more plausible than ω'. A possibility distribution allows to define two functions Π and N from the set of formulae over $\mathcal{L}$ to $[0,1]$ as follows:

$$\Pi(\varphi) := \max\{\pi(\omega) \mid \omega \in \Omega,\ \omega \models \varphi\} \text{ and } N(\varphi) := 1 - \Pi(\neg\varphi).$$

The function Π is called *possibility measure* and N is called *necessity measure*: $\Pi(\varphi)$ measures to what extent the formula φ is compatible with the available knowledge, while $N(\varphi)$ measures to what extent it is entailed. Given a possibility distribution π, a formula φ is said to be a *consequence* of π if and only if $\Pi(\varphi) > \Pi(\neg\varphi)$. Intuitively, φ is a consequence of π if the best models of φ (namely the models of φ having a highest degree) are more plausible (or more preferred) than the best models of $\neg\varphi$.

A *possibilistic formula* is a tuple $\langle \varphi, \alpha \rangle$ where $\varphi \in \mathcal{L}$ and $\alpha \in (0,1]$ expresses that φ is considered certain at least to the level α. Thus, given a necessity measure N, it holds $N(\varphi) \geq \alpha$. A *possibilistic knowledge base* is a set K of possibilistic formulas. The strict α-cut K_α of K is defined as $K_\alpha = \{\varphi \mid \langle \varphi, \beta \rangle \in K \textit{ and } \beta > \alpha\}$. From the strict α-cut, we define the inconsistency value of K $\mathit{Inc}(K) = \max\{\alpha \mid K_\alpha \text{ is inconsistent or } \alpha = 0\}$ which is the necessity degree under which information is ignored. We thus define $\mathit{Core}(K) = K_{\mathit{Inc}(K)}$ and say that a formula is a *consequence* of K (denoted by $K \vdash_\pi \varphi$) if $\mathit{Core}(K) \vdash \varphi$.

3 Possibilistic Logic Programs

We define the possibilistic extension of classic logic programs following [10]. These are logic programs where each rule is augmented with a necessity value.

A *possibilistic rule* is a pair $r = \langle r^*, \alpha \rangle$ where r^* is some rule in the sense of ASP and α denotes the rule's necessity degree in the range $(0,1]$. The ASP rule r^* is called the *projection* of the possibilistic rule r, i.e., the rule obtained by ignoring the attached necessity degree. A *possibilistic logic program* Q is a set of possibilistic rules. The projection of a possibilistic program Q^* is the set of rules $\{r^* \mid r \in Q\}$, i.e., a logic program in the sense of ASP.

Given a set A of propositional atoms, a *possibilistic atom* is a pair $p = \langle a, \alpha \rangle \in A \times (0, 1]$ where α denotes the necessity degree of a. A *possibilistic atom set* is a set of possibilistic atoms in which every propositional atom occurs in at most one possibilistic atom. Projections are defined as before, that is $p^* = a$ and $\{p_1, \dots\}^* = \{p_1^*, \dots\}$. Because stable models are sets of atoms, it seems intuitively appropriate to consider "possibilistic answer sets" of a possibilistic program as possibilistic atom sets. Indeed in [10] the projection of a possibilistic stable model of a possibilistic program is always a stable model of the projection of the program. However, in our opinion this is not adequate for reasons discussed in the following.

Before we introduce our new semantics of possibilistic logic programs we review the differnet semantics discussed in the literature.

Nicolas et al. [10]. Possibilistic answer sets are here defined as possibilistic atom sets whose projection is a classic answer set of the projection of the possibilistic program. The attached necessity values are determined according to the grounding sequence[1]. As a consequence, it does not capture all possible solutions of a possibilistic logic program with respect to possibilistic logic.

Example 1. Consider the possibilistic program $P_n = \{\langle \text{concert} \leftarrow \text{not canceled.}, 0.8 \rangle, \langle \text{canceled.}, 0.6 \rangle\}$. The only possibilistic answer set (in the sense of [10]) of this program is $\{\langle \text{canceled}, 0.6 \rangle\}$.

In this example the necessity value of *canceled* indicates that there exists some reason to believe it was not canceled. However, this is not reflected in the concept of possibilistic answer set by [10].

Moreover, Nicolas et al. define possibility distributions for possibilistic logic programs (although without providing an algorithm for computing it). The possibility of a set of atoms A given a logic program P is $\tilde{\pi}_P(A) = \pi_{P^A}(A)$ with $\pi_{P^A}(A)$ being defined as 0 if A is not grounded, 1 if it is a least model of the consequence operator applied to P^A, and $1 - \alpha$ otherwise (where α is the maximal possibility value among the non-satisfied rules).

Another issue with this semantics is the way how integrity constraints are treated. A violated integrity constraint prevents a set of atoms to be a possibilistic stable model independently of its necessity value, and a non-violated integrity constraint is ignored just as its necessity value. For example, the program $\{\langle \textit{concert} \leftarrow \text{not } \textit{canceled.}, 1 \rangle, \langle \bot \leftarrow \textit{concert.}, \alpha \rangle\}$ has no solution for any $\alpha \in (0, 1]$ even if α is very low.

Bauters et al. [9]. This approach starts from a different paradigm in order to overcome the issue described above. Bauters et al. identify that the possibilistic answer sets in the sense of Nicolas et al. [10] are computed based on the Gelfond-Lifschitz reduct of the program, which excludes any information about the necessity or possibility value

[1] A grounding sequence is the ordered set of rules involved in the deduction of an answer set.

that could have been drawn from the negative atoms. Bauters et al. also use possibilistic atom sets, but propose to consider and enforce the equalities $N(a) = \Pi(\text{not } a)$ and thus $N(\text{not } a) = 1 - N(a)$. In our opinion, this leads to some counter-intuitive results. Let us consider two examples to illustrate this (the first one has already been discussed by Bauters et al. [9]).

Example 2. Consider the possibilistic logic program: $P_{b1} = \{ \langle a \leftarrow \text{not } a., 1\rangle \}$. The only answer set in the sense of Bauters et al. [9] is $\{\langle a, 0.5\rangle\}$.

The rule $a \leftarrow$ not $a.$ used in this example is a self-contradiction which carries the maximal necessity value and, in the classic possibilistic logic context, would bring the inconsistency value of the program to 1.

Example 3. Consider the following possibilistic logic programs:

$$P_{b2} = \{ \langle a \leftarrow \text{not } a., 1\rangle, \langle b., 0.7\rangle \} \quad P_{b3} = \{ \langle a \leftarrow \text{not } a., 1\rangle, \langle b., 0.3\rangle \}$$

The only answer set for P_{b2} is $\{\langle a, 0.5\rangle, \langle b, 0.7\rangle\}$ and for P_{b3} $\{\langle a, 0.5\rangle, \langle b, 0.3\rangle\}$. The rank-ordering of the atoms a and b in the answer set is no longer only governed by the rank-ordering of the grounding sequence allowing their deduction. Here b is considered more certain than a if the necessity of the rule $b.$ is more than half the necessity of $a \leftarrow$ not a.

We think that the point made in the last example is in opposition to the philosophy of possibilistic logic where the necessity values only define a preordering between the formulas. It seems that the definition by Bauters et al. [9] introduces some logarithmic scale within the semantics which is not suitable from our point of view.

Bauters et al. [15]. The same authors proposed another approach for dealing with possibilistic logic and answer sets which is based on subsets of a program. This work differs from our approach in several aspects: first they consider every possible subset of the program which is harmful for the complexity and entails a different semantics. Also they do not consider answer sets properly speaking but only the brave and cautious consequences of the program. Moreover, this approach has a higher complexity, at the second level of the polynomial hierarchy [16].

Discussion. All of the approaches presented so far have problems. One of them is the inability to give an overall possibility value of the answer set as the possibilities are only attached to atoms. They are, for example, unable to measure the certainty of the empty set as a solution, e.g, to make the difference between $\langle a \leftarrow \text{not } a., 0.1\rangle$ and $\langle a \leftarrow \text{not } a., 1\rangle$. The semantics presented in the next section addresses these issues.

4 A New Definition of Possibilistic Answer Sets

In the examples presented previously, and given the non-monotonicity of the language, we can see that we could deduce atoms by ignoring the deduction of others. Ignoring rules as part of a definition can be helpful in two ways:

– It allows for considering more efficiently the reason why some atoms have been overlooked and to measure how harmful it was.
– It allows for defining possibilistic answer sets based on ι-answer sets.

As we want to represent the fact that some rules might have been ignored, we can introduce an overall possibility value for sets of atoms from which we define the concept of possibilistic interpretation. The possibility value is deduced from the necessity of rules going against the set of atoms.

Definition 5. *Let X be a possibilistic atom set and $\alpha \in (0,1]$. Then a* possibilistic interpretation *is a pair $\langle X, \alpha\rangle$ where α denotes the possibility degree of X.*

In this context, a possibilistic ι-answer set can be naturally defined with the help of an ι-answer set:

Definition 6. *Let P be a possibilistic logic program and $I = \langle X, \alpha\rangle$ be a possibilistic interpretation. Then I is a* possibilistic ι-answer set *of P if there exists some $Q \subseteq P$ such that the following holds:*

1. $X^* = CN^+(Q^*)$,
2. $body^+(Q^*) \subseteq CN^+(Q^*)$,
3. $body^-(Q^*) \cap CN^+(Q^*) = \emptyset$,
4. $\alpha = 1 - \max\{\beta \mid \langle r, \beta\rangle \in P \setminus Q\}$ *if* $P \neq Q$, *and* $\alpha = 1$ *otherwise,*
5. *for each* $a \in atoms(Q^*)$ *and* $r^* \in Q^*$ *such that* $a = head(r^*)$, $\{b_1, \ldots, b_m\} = body^+(r^*) \subseteq X$, *and* $body^-(r^*) \cap X = \emptyset$, *it holds* $N(a) \geq \min\{\alpha, \beta_1, \ldots, \beta_n\}$, *where* $N(r) = \alpha$ *and* $N(b_i) = \beta_i$ *and there exists some* $r^* \in Q^*$ *such that* $N(a) = \min\{\alpha, \beta_1, \ldots, \beta_n\}$,
6. *there is no* Q' *with* $Q \subsetneq Q' \subseteq P$ *satisfying conditions 2 to 4.*

On top of the necessity value of each atom, there is a possibility value for the overall possibilistic ι-answer set, which represents its plausibility as a solution. A possibilistic ι-answer set $I = \langle X, \alpha\rangle$ can thus be understood as follows: X can be accepted as a possibilistic ι-answer set but there exists a $1 - \alpha$ necessity against it; in case it is considered, the necessity of each atom is given in X.

With necessity being defined as a lower bound, classical possibilistic logic enforces formulas in a possibilistic knowledge base not to have a necessity degree of zero because otherwise they bring no information. For possibilistic ι-answer sets we enforce the same requirement.

Example 4. Consider the following possibilistic logic program:

$$P = \left\{ \begin{array}{ll} \langle rain \leftarrow \text{not } sun., 1\rangle, & \langle umbrella \leftarrow rain., 1\rangle, \\ \langle sun., 0.6\rangle, & \langle glasses \leftarrow sun., 1\rangle \end{array} \right\}$$

It has two possibilistic ι-answer set $\langle\{\langle sun, 0.6\rangle, \langle glasses, 0.6\rangle\}, 1\rangle$ with $P = Q$ and $\langle\{\langle rain, 1\rangle, \langle umbrella, 1\rangle\}, 0.4\rangle$ with $P \backslash Q = \{\langle sun., 0.6\rangle\}$.

A parallel can be established between possibilistic ι-answer sets and answer sets.

Proposition 1. *Let P be a possibilistic logic program and $I = \langle X, \alpha \rangle$ be a possibilistic ι-answer set of P. Then $\alpha = 1$ if and only if X^* is an answer set of P^* in the classical sense.*

The previous proposition follows directly from Theorem 3.6 of [12]. Moreover, this immediately shows the NP-hardness of finding possibilistic ι-answer sets. Here, the NP-membership is trivial as one has to guess the set of atoms X from Definition 6 and from there one can polynomially compute the consequence. From these observations, we obtain the following result.

Theorem 1. *Let P be a possibilistic logic program. Deciding the existence of an possibilistic ι-answer set for P is NP-complete.*

Integrity Constraints. In the original definition of ι-answer sets, Gebser et al. had to treat integrity constraints separately from the other rules. This distinction is necessary because discarding a rule from the applied rules set comes with no penalty in the original context. If there is no penalty in ignoring integrity constraints, they become useless. For our possibilistic ι-answer sets, such a special treatment is not necessary. The possibility value of an answer set reflects the importance of the discarded rules. We illustrate this in the following.

Example 5. Consider the following classic logic program and its possibilistic variant:

$$\begin{array}{l} P = \{a \leftarrow \text{not } b. \qquad b \leftarrow \text{not } a. \qquad \bot \leftarrow a.\} \\ P' = \{\langle a \leftarrow \text{not } b., 1\rangle \qquad \langle b \leftarrow \text{not } a., 1\rangle \qquad \langle \bot \leftarrow a., 0.5\rangle\} \end{array}$$

In P, there is one answer set here $\{b\}$ as $\{a\}$ is forbidden by the constraint. The constraint is considered part of the applied rules according to Definition 4. The necessity value attached to constraints can be used to rank-order the answer sets. Here we have, for example, $\langle\{\langle b, 1\rangle\}, 1\rangle$ and $\langle\{\langle a, 1\rangle\}, 0.5\rangle$ where the latter is considered less possible.

5 A Translation of Possibilistic Logic Programs into ASP

Gebser et al. [12] propose a translation, called ι-completion, from classical positive-order logic programs into SAT. We use this translation as a basis for encoding possibilistic logic programs into ASP.

Let P^* be a classical logic program, and P^*_C denote the set of self-blocking rules within P^* given by

$$P^*_C = \left\{r \mid r \in P^* \text{ and } \mathit{head}(r) \cap \mathit{body}^-(r) \neq \emptyset\right\}.$$

The set $\mathit{sup}(a)$ allows for identifying the necessary premisses to the deduction of a.

$$\begin{aligned} \mathit{rule}(a) &= \{r \in P^* \setminus P^*_C \mid \mathit{head}(r) = a\} \\ \mathit{sup}(a) &= \bigvee_{r \in \mathit{rule}(a)} \Big(\bigwedge_{p^+ \in body^+(r)} p^+ \wedge \bigwedge_{p^- \in body^-(r)} \neg p^- \Big) \end{aligned}$$

The set *block*(a) allows for identifying all the self blocking rules in which a is involved. *block*(a) is true if one self-blocking rule is supposed to be fired in the answer set, which is impossible. Thus, the ι-completion forces *block*(a) to be false.

$$neg(a) = \left\{r \in P^* \setminus P^*_C \mid a \in body^-(r)\right\}$$
$$block(a) = \bigvee_{r \in neg(a)} \left(head(r) \wedge \bigwedge_{p^+ \in body^+(r)} p^+ \wedge \bigwedge_{p^- \in body^-(r)} \neg p^-\right)$$

Let C^* denote the set of integrity constraints of P^*. The ι-completion is given by:

$$comp(P^*, C^*) = \{a \leftrightarrow sup(a) \wedge \neg block(a) \mid a \in atoms(P^*)\}$$

Because existing answer set solvers cannot handle floating point numbers, we assume in the following that both possibility and necessity values are given as integers in the range $\mathcal{V} = \{1, \ldots, 100\}$. For example, a necessity 0.8 is written as 80. This is not problematic as there are always only finitely many necessity values in a possibilistic program, and thus they can be accommodated on some finite integer scale.

The translation for ι-answer sets is done in four steps. The idea is close to the one introduced in [17]. The logic program checks for every possible interpretation whether it is a ι-answer set. After the checking is done, another part of the program finds the possibility value associated to the ι-answer set and to each of its atoms.

Step 1: Generating interpretations. In order to check every interpretation, we first have to assert that each atom is assigned true of false. To this end, we introduce for each atom $a \in atoms(P^*)$ an additional new atom na denoting not a, and the rules: $1\{l(a), l(na)\}1.$ and $\leftarrow l(a), l(na).$

Step 2: Checking for support. The second step is a translation of the completion given before. Namely, for each atom $a \in atoms(P^*)$ we introduce the rules:

$$l(a) \leftarrow sup(a), \text{not } block(a). \qquad l(na) \leftarrow \text{not } sup(a). \qquad l(na) \leftarrow block(a).$$

as well as the rules corresponding to $sup(a)$ and $block(a)$. They can easily be translated into ASP using labeling conversion.

Step 3: Computing necessity values. The third step computes the consequence of the reduct. An atom must be deduced under two conditions: it needs to have a rule allowing its deduction, and the atoms in this rule's positive body must be justified. To achieve this, we introduce the following for each rule $\langle r^*, v\rangle \in P$ and for each $N \leq v \in \mathcal{V}$.

$$vlip(head(r^*), N) \leftarrow l(head(r^*)), vlip(body^+(r^*), N), \text{not } body^-(r^*), N \leq v.$$

Here N represents the necessity value of the atom which has to be equal to the minimum amongst the necessity value of the atoms in the positive body and the rule itself. Thus, the head is deduced with a necessity v if all atoms in $body^+(r^*)$ have been deduced with at least a necessity v and the necessity of the rule is also at least v.

We need to compute the final necessity value for each of the atoms. For each $p \in atoms(P^*), \forall N, O \in \mathcal{V}$ we introduce the rules where L is a variable representing atoms.

$$negvli(L, N) \leftarrow vlip(L, N), vlip(L, O), N < O.$$
$$vli(L, N) \leftarrow vlip(L, N), \text{not } negvli(L, N).$$

Step 4: Computing the possibility value. For the possibility value, we need first to check rules that might have been ignored. We introduce the rule:

$$\mathit{vasp}(100 - v) \leftarrow \text{not } \mathit{head}(r^*), \mathit{body}^+(r^*), \text{not } \mathit{body}^-(r^*).$$

for each rule $\langle r^*, v\rangle \in P$. From all the rules ignored, the higher necessity value is the most relevant (and thus the smaller possibility $\mathit{vasp}(N)$). The first rule marks the values which are not minimal. Then the minimal value is the one that remains unmarked as computed by the second rule. For each $N \in \mathcal{V} \cup \{0\}$ this is achieved by the two rules: $\mathit{negvas}(N) \leftarrow \mathit{vasp}(N), \mathit{vasp}(O), N > O.$ $\quad \mathit{vas}(N) \leftarrow \mathit{vasp}(N), \text{not } \mathit{negvas}(N).$ All these 4 steps together form the program $\tau(P)$.

Theorem 2. *The stable models of the translated program $\tau(P)$ are exactly the possibilistic ι-answer sets of P.*

The proof comes from the separation of the program in two parts. The first part (step 1 and 2) is exactly the translation proposed in [12] and the second part (step 3 and 4) cannot prevent a set to be a solution, but only computes the necessity and possibility values associated.

Example 6. We present an example to illustrate the translation process. Let P be the following possibilistic logic program:

$$P = \left\{ \begin{array}{ccc} \langle a \leftarrow \text{not } b., 1\rangle & \langle d \leftarrow a., 1\rangle & \langle b \leftarrow \text{not } c., 0.8\rangle \\ \langle e \leftarrow b., 1\rangle & \langle c \leftarrow \text{not } a., 0.6\rangle & \langle f \leftarrow c., 1\rangle \end{array} \right\}$$

Program P leads to the final translation in ASP presented in Figure 1[2]. For the sake of space, the *sup* and *neg* rules are given only for the atom a. Only the atoms of the solutions which are relevant for understanding are given. This translation has 3 solutions partially exhibited here:

$$\left\{ \begin{array}{l} \{l(a), l(nb), l(nc), l(d), l(ne), l(nf), vas(20), vli(d, 100), vli(a, 100)\}, \\ \{l(na), l(b), l(nc), l(nd), l(e), l(nf), vas(40), vli(e, 80), vli(b, 80)\}, \\ \{l(na), l(nb), l(c), l(nd), l(ne), l(f), vas(0), vli(f, 60), vli(c, 60)\} \end{array} \right\}$$

The last solution should be ignored because its possibility is equal to zero.

Experimental Results. In order to evaluate the usability of our approach we ran a series of tests[3] with clingo [18] on an Intel Pentium with 2 GHz. The tests were performed on randomly generated instances with two parameters: the number of rules *nbr* and the number of atoms *nba*. The average running times over 1000 instances for a pair (*nbr*,*nba*) were (500,250) in 0.34s, (1000,500) in 0.97s, (5000,2500) in 15.84s and (10000,5000) in 58.03s. For comparison, a normal ASP (10000,5000) instance is solved in 0.280s on average. This suggests that finding possibilistic ι-answer sets can be performed on instances of acceptable size.

[2] The statement $1\{a_1, ..., a_n\}1$ stands for "exactly one atom in $\{a_1, ..., a_n\}$ is true".

[3] The script used is available at `http://www.informatik.uni-freiburg.de/ hue/translate.tar.gz`

$$
\begin{array}{llll}
1\{l(a), l(na)\}1 & 1\{l(b), l(nb)\}1 & 1\{l(c), l(nc)\}1 & 1\{l(d), l(nd)\}1 \\
 & 1\{l(e), l(ne)\}1 & 1\{l(f), l(nf)\}1 &
\end{array}
$$

$$
\begin{array}{ll}
negvli(L, N) \leftarrow vlip(L, N), vlip(L, O), N < O. & vli(L, N) \leftarrow vlip(L, N), \text{not } negvli(L, N). \\
vlip(a, N) \leftarrow l(a), l(nb), N <= 100. & vlip(b, N) \leftarrow l(b), l(nc), N <= 80. \\
vlip(c, N) \leftarrow l(c), l(na), N <= 60. & vlip(d, N) \leftarrow l(d), vlip(a, N), N <= 100. \\
vlip(e, N) \leftarrow l(e), vlip(b, N), N <= 100. & vlip(f, N) \leftarrow l(f), vlip(c, N), N <= 100. \\
negvas(N) \leftarrow vasp(N), vasp(O), N > O. & vas(N) \leftarrow vasp(N), \text{not } negvas(N).
\end{array}
$$

$$
\begin{array}{lll}
vasp(0) \leftarrow l(na), l(nb). & vasp(20) \leftarrow l(nb), l(nc). & vasp(40) \leftarrow l(nc), l(na). \\
vasp(0) \leftarrow l(nd), l(a). & vasp(0) \leftarrow l(ne), l(b). & vasp(0) \leftarrow l(nf), l(c). \\
sup(a) \leftarrow suprule(a, r_0). & suprule(a, r_0) \leftarrow l(nb). & neg(a) \leftarrow negrule(a, r_0). \\
negrule(a, r_0) \leftarrow l(c), l(na). & l(a) \leftarrow sup(a), \text{not } neg(a). & l(na) \leftarrow \text{not } sup(a). \\
 & l(na) \leftarrow neg(a). &
\end{array}
$$

Fig. 1. ASP translation of Example 6

6 Conclusion

In this paper we considered an extension of ASP that involves possibilistic rules. Such possibilistic logic programs allows for concisely expressing degrees of possibility and giving a rank-ordering of the program's rules. While the concept of possibilistic logic programs has been considered before, previous semantics of the resulting possibilistic answer sets are in our opinion contrary to the intuition behind possibilistic logic and lack several expected properties.

We have here provided a new semantics for possibilistic programs based on the existing concept of ι-answer sets. Our semantics extends the original definition by Nicolas et al. [10], overcoming its shortcomings and providing a reasonable concept of possibilistic solution that is not limited to the classic stable models of a logic program. Moreover, our definition handles inconsistencies and integrity constraints much more gracefully than alternative suggestions put forward by Bauters et al. [9]. The new semantics is in line with and respects the original philosophy of possibilistic logic in the sense of rank-ordering rules. This is in contrast to Bauters et al. [9] who require computations on necessity values exceeding a pure rank-ordering.

Our definition of possibilistic answer sets allows for computing them using existing answer set tools by transforming possibilistic logic programs into classic logic programs. Thus, possibilistic rules can be easily applied in existing application scenarios based on modeling only without requiring new tools.

Acknowledgements. We thank the anonymous reviewers of this paper and a previous version for helpful suggestions and comments. J.H. and M.W. acknowledge support by the DFG Transregional Collaborative Research Center *SFB/TR 8 Spatial Cognition* (project: R4-[LogoSpace]). S.W. acknowledges support by the DFG priority program *SPP 1516: New Frameworks of Rationality* (project: Models of human non-monotonic reasoning).

References

1. Lifschitz, V.: Thirteen definitions of a stable model. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Fields of Logic and Computation. LNCS, vol. 6300, pp. 488–503. Springer, Heidelberg (2010)
2. Brewka, G., Eiter, T.: Preferred answer sets for extended logic programs. Artif. Intell. 109(1-2), 297–356 (1999)
3. Bacchus, F., Grove, A.J., Halpern, J.Y., Koller, D.: From statistical knowledge bases to degrees of belief. Artificial Intelligence 87(1-2), 75–143 (1996)
4. Thimm, M., Kern-Isberner, G.: On probabilistic inference in relational conditional logics. Logic Journal of the IGPL 20(5), 872–908 (2012)
5. Poole, D.: Logic programming, abduction and probability. In: Proceedings of the International Conference on Fifth Generation Computer Systems (FCGS), pp. 530–538 (1992)
6. Lukasiewicz, T.: Probabilistic logic programming. In: ECAI, pp. 388–392 (1998)
7. Baral, C., Gelfond, M., Rushton, J.N.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming 9(1), 57–144 (2009)
8. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 439–513 (1994)
9. Bauters, K., Schockaert, S., De Cock, M., Vermeir, D.: Possibilistic answer set programming revisited. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI). AUAI Press (2010)
10. Nicolas, P., Garcia, L., Stéphan, I.: Possibilistic stable models. In: IJCAI, pp. 248–253. Morgan Kaufmann Publishers (2005)
11. Osorio, M., Nieves, J.C.: PStable semantics for possibilistic logic programs. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 294–304. Springer, Heidelberg (2007)
12. Gebser, M., Gharib, M., Mercer, R.E., Schaub, T.: Monotonic answer set programming. Journal of Logic and Computation 19(4), 539–564 (2009)
13. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference on Logic Programming (ICLP), pp. 1070–1080 (1988)
14. Baral, C., Kreinovich, V., Lifschitz, V.: Introduction: Logic programming, non-monotonic reasoning and reasoning about actions. Annals of Mathematics and Artificial Intelligence 21(2-4), 129 (1997)
15. Bauters, K., Schockaert, S., Cock, M.D., Vermeir, D.: Possible and necessary answer sets of possibilistic answer set programs. In: ICTAI, pp. 836–843. IEEE (2012)
16. Bauters, K., Schockaert, S., De Cock, M., Vermeir, D.: Answer set programs with optional rules: a possibilistic approach. Working papers of the IJCAI 2013 Workshop on Weighted Logics for Artificial Intelligence, WL4AI 2013, pp. 2–9 (2013)
17. Hué, J., Papini, O., Würbel, E.: Removed sets fusion: Performing off the shelf. In: ECAI, pp. 94–98 (2008)
18. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AICOM 24(2), 107–124 (2011)