Weak and strong disjunction in possibilistic ASP

Weak and strong disjunction in possibilistic ASP

Kim Bauters[1⋆], Steven Schockaert[1⋆⋆], Martine De Cock[1], and Dirk Vermeir[2]

[1] Department of Applied Mathematics and Computer Science
Universiteit Gent, Krijgslaan 281, 9000 Gent, Belgium
(kim.bauters, steven.schockaert, martine.decock)@ugent.be
[2] Department of Computer Science
Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium
dvermeir@vub.ac.be

Abstract. Possibilistic answer set programming (PASP) unites answer set programming (ASP) and possibilistic logic (PL) by associating certainty values with rules. The resulting framework allows to combine both non-monotonic reasoning and reasoning under uncertainty in a single framework. While PASP has been well-studied for possibilistic definite and possibilistic normal programs, we argue that the current semantics of possibilistic disjunctive programs are not entirely satisfactory. The problem is twofold. First, the treatment of negation-as-failure in existing approaches follows an all-or-nothing scheme that is hard to match with the graded notion of proof underlying PASP. Second, we advocate that the notion of disjunction can be interpreted in several ways. In particular, in addition to the view of ordinary ASP where disjunctions are used to induce a non-deterministic choice, the possibilistic setting naturally leads to a more epistemic view of disjunction. In this paper, we propose a semantics for possibilistic disjunctive programs, discussing both views on disjunction. Extending our earlier work, we interpret such programs as sets of constraints on possibility distributions, whose least specific solutions correspond to answer sets.

1 Introduction

Answer Set Programming (ASP) is a form of declarative programming based on the stable model semantics [11] that allows to succinctly formulate and easily solve complex combinatorial problems. Possibilistic logic (PL) [7], which is based on possibility theory [17], allows us to reason about (partial) ignorance or uncertainty in a non-probabilistic way. Possibilistic ASP (PASP) [13, 3] unites ASP and PL and provides a single framework for declarative programming under uncertainty. The certainty of a conclusion is then given by the lowest certainty of the rules that were used to establish the conclusion (*i.e.* the strength of the conclusion is determined by the weakest piece of information involved).

The semantics of both ordinary and possibilistic ASP can be characterized as constraints on possibility distributions [3]. Under this characterization we treat

⋆ Funded by a joint Research Foundation-Flanders (FWO) project
⋆⋆ Postdoctoral fellow of the Research Foundation-Flanders (FWO)

a rule of the form '$c \leftarrow a, not\ b$' intuitively as follows: we can conclude that 'c' is certain when we know that 'a' is certain and when it is consistent to assume that 'b' is false. This characterization of ASP clearly highlights the intuition that underlies ASP and the epistemic flavor of such rules. Indeed, when we know that 'a' is true and we do not know that 'b' is true then we conclude that 'c' should be accepted as true. In the possibilistic case, when we attach certainty degrees to the atoms and rules, we have that the certainty of the conclusion can be no stronger than the certainty of the different pieces of information that were used to deduce the conclusion.

When we consider disjunctions in the head of rules, then reasoning under this epistemic view suggests that when we are certain of the body, we should accept the head of the rule. For example, given the rules

$$\begin{aligned} a \vee b \leftarrow & \\ c \leftarrow a & \\ c \leftarrow b & \end{aligned}$$

the epistemic reading of the program is that we know that '$a \vee b$' is true and that as soon as we know explicitly whether it is 'a' or 'b' that is true we can conclude c. Hence, without any further information, we cannot conclude 'c' since we only have the underspecified information that '$a \vee b$' is true. We would be able to conclude 'c', however, if we had a rule $c \leftarrow a \vee b$. This particular view of disjunction does not correspond with the intuition in ordinary ASP. Indeed, the semantics of disjunctive rules in ASP say that whenever the body of a rule is satisfied, we should make a non-deterministic choice as to which atom in the head of our rule is chosen to be true (alongside with a minimality requirement on the resulting answer sets). Regardless, as we will see it is often the case that when reasoning under uncertainty we are driven towards this epistemic view of disjunction.

In this paper we examine the differences between these two treatments of disjunction in the head using the framework of possibilistic logic. As we will see, if we treat ASP rules as constraints on possibility distributions we naturally obtain two ways in which we can interpret a disjunctive rule. In one case we retrieve the semantics of ordinary disjunctive ASP (this interpretation of disjunction will be called strong disjunction) and in the other case we retrieve the epistemic view of disjunction (this interpretation of disjunction will be called weak disjunction). The resulting characterizations of disjunctive ASP programs can then naturally be generalized to possibilistic programs, where each rule is labelled with a degree of certainty.

The remainder of this paper is organized as follows. In Section 2 we start by introducing some background on ASP, PL and PASP. In Section 3 we present the strong semantics for possibilistic disjunctive ASP. Then in Section 4 we present the weak semantics for possibilistic disjunctive ASP. We discuss related work in Section 5 and we conclude with Section 6 in which we provide our conclusions.

2 Preliminaries

2.1 Answer Set Programming

To define ASP programs, we start from a finite set of atoms $\mathcal{A}$. A *naf-atom* is either an atom 'a' or an atom 'a' preceded by '*not*' which we call the *negation-as-failure operator*. Intuitively, '*not* a' is true when we cannot prove 'a'.

An expression of the form $a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n$ with a_i an atom with $0 \leq i \leq n$ is called a *disjunctive rule*. We call $a_0; ...; a_k$ the *head* of the rule (interpreted as a disjunction) and $a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n$ the *body* of the rule (interpreted as a conjunction).

A positive disjunctive rule is a disjunctive rule without negation-as-failure, *i.e.* $n = m$. A rule of the form $a_0; ...; a_k \leftarrow$ is called a *fact* and is used as a shorthand for $a_0; ...; a_k \leftarrow \top$ with $\top$ a special language construct that denotes tautology.

A disjunctive program P is a finite set of disjunctive rules. The *Herbrand base* $\mathcal{B}_P$ of a disjunctive program P is the set of atoms appearing in P. An *interpretation* I of a disjunctive program P is any set of atoms $I \subseteq \mathcal{B}_P$. A *normal rule* is a disjunctive rule with exactly one atom in the head, *i.e.* $k = 0$. A *definite rule* is a normal rule with no negation-as-failure, *i.e.* $k = 0$ and $n = m$. A *normal (*resp. *definite) program* P is a finite set of normal (*resp.* definite) rules.

An interpretation I is a *model* of a positive disjunctive rule $r = a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m$, denoted $I \models r$, if $\{a_0, ..., a_k\} \cap I \neq \emptyset$ or $\{a_{k+1}, ..., a_m\} \not\subseteq I$, *i.e.* the body is false or at least one of the atoms in the head is true. An interpretation I of a positive disjunctive program P is a *model* of P iff $\forall r \in P \cdot I \models r$.

The *reduct* [11, 10] P^I of a disjunctive program P *w.r.t.* an interpretation I is defined as

$$\begin{aligned} P^I = \{a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m \ \mid\ & (\{a_{m+1}, ..., a_n\} \cap I = \emptyset) \\ & \wedge (a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n) \in P\}. \end{aligned}$$

We say that I is an answer set of the disjunctive program P when I is a minimal model *w.r.t.* set inclusion of P^I.

The answer set of a definite program P can also be defined using the *immediate consequence operator* T_P, which is defined *w.r.t.* an interpretation I as:

$$T_P(I) = \{a_0 \mid (a_0 \leftarrow a_1, ..., a_m) \in P \wedge \{a_1, ..., a_m\} \subseteq I\}.$$

We use $P^\star$ to denote the fixpoint which is obtained by repeatedly applying T_P starting from the empty interpretation, *i.e.* the least fixpoint of T_P *w.r.t.* set inclusion, which is guaranteed to exist [16]. An interpretation I is an *answer set of a definite program* P iff $I = P^\star$.

2.2 Possibilistic Logic

At the semantic level, possibilistic logic [7] is defined in terms of a *possibility distribution* π on the universe of interpretations. For $\Omega = 2^{\mathcal{B}_P}$ the set of all

interpretations of a program P, we have that the possibility distribution is an $\Omega \to [0,1]$ mapping which encodes for each interpretation (or world) I to what extent it is plausible that I is the actual world. Rather than using certainty degrees from $[0,1]$, we could use any linearly ordered set, together with an involutive order-reversing mapping. Intuitively, $\pi(I)$ represents the compatibility of the interpretation I with available information. By convention, $\pi(I) = 0$ means that I is impossible and $\pi(I) = 1$ means that no available information prevents I from being the actual world. Note that possibility degrees are mainly interpreted qualitatively: when $\pi(I) > \pi(I')$, I is considered more plausible than I'. For two possibility distributions π_1 and π_2 with the same domain Ω we write $\pi_1 > \pi_2$ when $\forall I \in \Omega \cdot \pi_1(I) \geq \pi_2(I)$ and $\exists I \in \Omega \cdot \pi_1(I) > \pi_2(I)$. The satisfaction relation $\models$ is defined for a set of atoms A as $A \models a$ iff $a = \top$ or $a \in A$, otherwise $A \not\models a$. Furthermore, $A \models \neg a$ iff $A \not\models a$.

A possibility distribution π induces two uncertainty measures that allow us to rank propositions. The *possibility measure* Π is defined by [7]:

$$\Pi(p) = \max\{\pi(I) \mid I \models p\}$$

and evaluates the extent to which a proposition p is consistent with the beliefs expressed by π. The dual *necessity measure* N is defined by:

$$N(p) = 1 - \Pi(\neg p)$$

and evaluates to which extent a proposition p is entailed by available beliefs [7].

An important property that necessity measures have is min-decomposability *w.r.t.* conjunction: $N(p \wedge q) = \min(N(p), N(q))$ for all propositions p and q. However, for disjunction only the inequality $N(p \vee q) \geq \max(N(p), N(q))$ holds. As possibility measures are dual to necessity measures, they have the important property of max-decomposability *w.r.t.* disjunction, whereas for conjunction only the inequality $\Pi(p \wedge q) \leq \min(\Pi(p), \Pi(q))$ holds.

At the syntactic level, a *possibilistic knowledge base* Σ corresponds to a set of constraints $N(p) \geq c$ where p is a propositional formula and $c \in [0,1]$ expresses the certainty that p is the case. Typically, there will be many possibility distributions that satisfy these constraints. In practice, we are usually only interested in the *least specific possibility distribution* of these possibility distributions, which is the possibility distribution that makes minimal commitments, *i.e.* the largest possibility distribution *w.r.t.* the ordering $>$ defined above.

2.3 Possibilistic normal ASP

Possibilistic ASP combines ASP and possibilistic logic [7] by associating a certainty value with atoms and rules. A possibilistic normal (*resp.* definite) rule is a pair (r, λ) where r is a normal (*resp.* definite) rule and where $\lambda \in [0,1]$ is a certainty attached to r. We also write a pair (r, λ) as 'λ**:** r'. A possibilistic normal (*resp.* definite) program is a set of possibilistic normal (*resp.* definite) rules.

As we recalled in Section 2.1, in ASP, an answer set of a program P is an interpretation that satisfies some additional requirements. Note that an interpretation I of P can be thought of as a $\mathcal{B}_P \to \{0,1\}$ mapping. As a generalization

of this, in possibilistic ASP, an answer set of a program is a valuation V, which is a $\mathcal{B}_P \to [0,1]$ mapping, that satisfies the requirements formally defined in Definition 1. This is the mechanism used to associate certainty values with atoms appearing in a program. The intuition is that for an atom $a \in \mathcal{B}_P$, $V(a) = c$ means that we can derive with certainty c that a is true. For notational convenience, we also use the set notation $V = \{a^c, \ldots\}$. In accordance with this set notation, we write $V = \emptyset$ to denote the valuation in which each atom is mapped to 0. We generally omit atoms and rules with an associated certainty of 0 due to their triviality.

Possibilistic normal programs (and therefore also ordinary normal programs) are interpreted in terms of constraints on possibility distributions. Intuitively, an ordinary rule of the form '$rule = (head \leftarrow body)$' says that we are able to conclude that '*head*' is true when we know that '*body*' is true. When we associate necessities with the information in '*body*' and with '*rule*' itself, then we can only deduce '*head*' with a certainty $\min\{N(body), N(rule)\}$. Indeed, the contribution of a single rule to the necessity with which its head is true cannot be stronger than the necessity of the weakest information used to derive the body of the rule. However, a stronger conclusion could be derived using another set of rules, hence the '$rule = (head \leftarrow body)$' induces the constraint $N(head) \geq \min\{N(body), N(rule)\}$.

In the ordinary case, we tackle negation-as-failure by making certain assumptions about which atoms we will be able to derive (we guess an I) and then checking whether our assumptions are stable (we verify that $I = (P^I)^\star$). When dealing with uncertain information, the assumptions we need to make are not whether an atom is true or not, but rather with what certainty we will be able to derive an atom. We make these assumptions by guessing a valuation V, *i.e.* an association of a necessity with each atom. At the end we verify whether $V(a) = N(a)$, *i.e.* whether our guess is stable. This is the possibilistic counterpart of the ordinary reduct.

Definition 1. *[3] Let P be a possibilistic normal program. Let $V : \mathcal{B}_P \to [0,1]$ be a valuation. For every $p \in P$, the constraint $\gamma_V(p)$ induced by $p = (r, \lambda)$ with $r = (a_0 \leftarrow a_1, ..., a_m, not\ a_{m+1}, ..., not\ a_n)$ and V is given by*

$$N(a_0) \geq \min\{N(a_1), ..., N(a_m), 1 - V(a_{m+1}), ..., 1 - V(a_n), \lambda\}.$$

We write $C_{(P,V)} = \{\gamma_V(p) \mid p \in P\}$ to denote the set of constraints imposed by program P. A possibility distribution that satisfies the constraints in $C_{(P,V)}$ is called a possibilistic model of $C_{(P,V)}$. We write $S_{(P,V)}$ for the set of all least specific possibilistic models of $C_{(P,V)}$. V is called a possibilistic answer set of P iff there exists a $\pi \in S_{(P,V)}$ such that $\forall a \in \mathcal{B}_P \cdot V(a) = N(a)$.

The ordinary case can be retrieved if we also require $\forall a \in \mathcal{B}_P \cdot N(a) \in \{0,1\}$, *i.e.* if for every atom we are entirely sure whether or not the atom is necessary. The above definitions can then be used to characterize the semantics of ordinary normal programs.

Proposition 1. *[3] Let P' be a normal program, let P be a possibilistic normal program such that $P = \{(r', 1) \mid r' \in P'\}$ and let $V : \mathcal{B}_P \to [0, 1]$ be a valuation. If V is a possibilistic answer set of P and $\forall a \in \mathcal{B}_P \cdot V(a) \in \{0, 1\}$, then $M = \{a \mid V(a) = 1, a \in \mathcal{B}_P\}$ is an answer set of P'.*

Before we extend the semantics to cover the case of disjunction in Section 3 and 4, we first provide an example of the possibilistic semantics applied to a PASP program in order to further clarify the approach.

Example 1. Two common symptoms associated with fibromyalgia (a medical disorder consisting of pain in muscle and joint tissue) are a feeling of weakness and joint pain, where feeling weak without other causes is a telltale sign of fibromyalgia. Our patient tells us that she is experiencing both symptoms. However, the patient is known as a hypochondriac and is not entirely trustworthy. In the past she sometimes complained about weakness without any grounds, though she hardly ever complains about pain without an actual physical or mental cause. Her sagging eyes hint at an iron deficiency (which might explain the weakness in itself), though it is highly unlikely that sagging eyes by themselves correctly identify an iron deficiency. We have the program P with the rules:

$$\begin{aligned}
&\mathbf{0.2:}\ \mathit{fibro} \leftarrow \mathit{pain}\\
&\mathbf{0.6:}\ \mathit{fibro} \leftarrow \mathit{weak}, \mathit{not}\ \mathit{deficiency}\\
&\mathbf{0.9:}\ \mathit{pain} \leftarrow\\
&\mathbf{0.8:}\ \mathit{weak} \leftarrow\\
&\mathbf{0.1:}\ \mathit{deficiency} \leftarrow
\end{aligned}$$

which induces the set $C_{(P,V)}$ of constraints

$$\begin{aligned}
\{&N(\mathit{fibro}) \geq \min\{N(\mathit{pain}), 0.2\},\\
&N(\mathit{fibro}) \geq \min\{N(\mathit{weak}), 1 - V(\mathit{deficiency}), 0.6\},\\
&N(\mathit{pain}) \geq 0.9 \quad , \quad N(\mathit{weak}) \geq 0.8 \quad , \quad N(\mathit{deficiency}) \geq 0.1\}.
\end{aligned}$$

The set of least specific possibility models $S_{(P,V)}$ is a singleton and $\pi \in S_{(P,V)}$ is defined as $\pi(I) = 0.1$ when $I \models \{\neg p\}$, $\pi(I) = 0.2$ when $I \models \{p, \neg w\}$, $\pi(I) = 0.4$ when $I \models \{\neg f, p, w\}$, $\pi(I) = 0.9$ when $I \models \{f, p, w, \neg d\}$ and $\pi(I) = 1$ when $I \models \{f, p, w, d\}$, where we use the first letter of the atom as abbreviation to save space. The possibilistic answer set of this program is unique and is given by

$$V = \left\{\mathit{pain}^{0.9}, \mathit{weak}^{0.8}, \mathit{deficiency}^{0.1}, \mathit{fibro}^{0.6}\right\}$$

which can readily be verified.

We would also like to point out that, unlike the approach above, the approaches from [13, 14] do not take the *extent* of certainty of information into account when determining the reduct of a PASP program. In these other semantics, *any* proof of 'a', no matter how uncertain it is, suffices to eliminate

the expression '*not a*'. Hence, in the example above, rule 2 would be eliminated based on rule 5, and we would only be able to derive $\mathit{fibro}^{0.2}$. This clearly is not the intended meaning of the program as the limited certainty of an actual deficiency should not be sufficient to dismiss the certainty we have in diagnosing fibromyalgia.

3 Strong Possibilistic Semantics

In this section we extend the semantics of possibilistic normal ASP [3] to disjunctive programs, in a way which remains faithful both to ordinary disjunctive ASP (see Section 2.1) and to the semantics from [6]. As necessity measures do not have the max-decomposability property, we have a choice of how to interpret the disjunction in the head. This is similar to the choice one has for the semantics of disjunction when characterizing ASP using autoepistemic logic [12] or using meta-rules in possibilistic logic [9]. A *possibilistic disjunctive rule* $p = (r, \lambda)$ with $r = a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m, \mathit{not}\ a_{m+1}, ..., \mathit{not}\ a_n$ can either be interpreted as the constraint

$$\max\{N(a_0), ..., N(a_k)\} \geq \min\{N(a_{k+1}), ..., N(a_m), 1 - V(a_{m+1}), ..., 1 - V(a_n), \lambda\} \quad (1)$$

which we will call *strong disjunction* or as the constraint

$$N(a_0 \vee ... \vee a_k) \geq \min\{N(a_{k+1}), ..., N(a_m), 1 - V(a_{m+1}), ..., 1 - V(a_n), \lambda\} \quad (2)$$

which we will call *weak disjunction*. This is a choice that does not arise for the conjunction in the body since min-decomposability dictates that $N(a \wedge \ldots \wedge z) = \min\{N(a), \ldots, N(z)\}$. However, for the disjunction, we only have $N(a \vee \ldots \vee z) \geq \max\{N(a), \ldots, N(z)\}$.

The choice of how to treat disjunction is an important one that profoundly impacts the nature of the resulting answer sets. The main distinction between strong and weak disjunction has to do with the way that we regard an answer set. If we see an answer set as a solution to a problem, then the non-deterministic nature of strong disjunction provides a useful way to generate different (candidate) solutions. If we take an answer set as a representation of an epistemic state, then weak disjunction models the current state of belief.

In the remainder of this section we consider the characterization of disjunction as (1). In Section 4 we consider the characterization of disjunction as (2).

As it turns out, the characterization of disjunction as (1) makes the disjunction behave as in ordinary ASP (see Section 2.1). Indeed, the interplay between the strong possibilistic semantics of disjunction together with the requirement that we are looking for the least specific possibility distribution ensures that disjunction induces a choice. Similar as in the ordinary case, the constraint (1) will generate a number of possible outcomes. The requirement that we are looking for the least specific possibility distribution behaves similarly as the requirement of trying to find the minimal model. We will first give the general definition and then we will illustrate the semantics using an ordinary disjunctive ASP program.

Definition 2. *Let P be a possibilistic disjunctive program and let $V : \mathcal{B}_P \to [0,1]$ be a valuation. For every $p \in P$ with $p = (r, \lambda)$, the constraint γ^{s}_V induced by $r = (a_0; ...; a_k \leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n)$ and V under the strong possibilistic semantics of disjunction is given by*

$$\max\{N(a_0), ..., N(a_k)\} \geq \min\{N(a_{k+1}), ..., N(a_m), \\ 1 - V(a_{m+1}), ..., 1 - V(a_n), \lambda\}.$$

$C^{\mathrm{s}}_{(P,V)} = \{\gamma^{\mathrm{s}}_V(r) \mid r \in P\}$ is the set of constraints imposed by program P and V. A possibility distribution that satisfies the constraints in $C^{\mathrm{s}}_{(P,V)}$ is called a possibilistic model of $C^{\mathrm{s}}_{(P,V)}$. We write $S^{\mathrm{s}}_{(P,V)}$ to denote the set of all least specific possibilistic models of $C^{\mathrm{s}}_{(P,V)}$. V is called a possibilistic answer set of P iff there exists a $\pi \in S^{\mathrm{s}}_{(P,V)}$ such that $\forall a \in \mathcal{B}_P \cdot V(a) = N(a)$.

Whenever P is a positive disjunctive program we have no need for a specific valuation V to pin down the constraints (since there is no negation-as-failure), and we simplify the notation to $\gamma^{\mathrm{s}}, C^{\mathrm{s}}_P$ and S^{s}_P. As before we retrieve the semantics for the ordinary case if we require $\forall a \in \mathcal{B}_P \cdot N(a) \in \{0, 1\}$.

Example 2. Consider the program P with the single rule

$$\mathbf{0.7:}\ a; b \leftarrow .$$

The set of constraints C^{s}_P is given by $\{\max\{N(a), N(b)\} \geq 0.7\}$. This constraint induces a choice, *i.e.* we either need to pick $N(a) = 0.7$ or $N(b) = 0.7$ to conform to the principle of least specificity. We can conclude that the two possibilistic answer sets of P are given by $\{a^{0.7}\}$ and $\{b^{0.7}\}$. This corresponds with the non-deterministic intuition of the problem; we choose either 'a' or 'b' and assign a certainty of 0.7 to the atom we choose.

Example 3. We cannot simply simulate disjunction using negation-as-failure, as would be possible in the ordinary case. Indeed, the possibilistic normal program that simulates program P from Example 2 would have the rules

$$\mathbf{0.7:}\ a \leftarrow not\ b \qquad\qquad \mathbf{0.7:}\ b \leftarrow not\ a.$$

This program has an infinite set of possibilistic answer sets, with certainty degrees ranging from 0.3 to 0.7 for a and $1 - N(a)$ for b. When trying to simulate the rule from Example 2 we clearly do not want a possibilistic answer set such as $\{a^{0.4}, b^{0.6}\}$ as this does not correspond with our intuition. This again highlights the importance of satisfactory semantics for possibilistic disjunctive ASP.

As before, when we impose the additional constraint $\forall a \in \mathcal{B}_P \cdot N(a) \in \{0, 1\}$ we retrieve the ordinary semantics for disjunctive ASP.

Proposition 2. *Let P' be a disjunctive program, let P be a possibilistic disjunctive program such that $P = \{(r', 1) \mid r' \in P'\}$. Let $V : \mathcal{B}_P \to [0,1]$ be a valuation. If V is a possibilistic answer set of P and $\forall a \in \mathcal{B}_P \cdot V(a) \in 0, 1$, then $M = \{a \mid V(a) = 1, a \in \mathcal{B}_P\}$ is an answer set of the disjunctive program P'.*

Example 4. Consider the possibilistic disjunctive program P with the rules

$$\mathbf{1:}\ a; b \leftarrow \qquad\qquad \mathbf{1:}\ a \leftarrow b.$$

which induces the constraints

$$\max\{N(a), N(b)\} \geq N(\top) = 1 \qquad\qquad N(a) \geq N(b).$$

Intuitively, the first constraint induces a choice. To satisfy the constraint, we need to take either $N(a) = 1$ or $N(b) = 1$. Since $N(a) = 1 = 1 - \Pi(\neg a)$ tells us that $\Pi(\neg a) = 0 = \max\{\pi(I) \mid I \models \neg a\}$, we have for every I with $a \notin I$ that $\pi(I) = 0$ (and obviously when $N(b) = 1$ then $\pi(I) = 0$ whenever $b \notin I$). Depending on our choice of whether we take $N(a) = 1$ or $N(b) = 1$ (and since $N(a) = N(b)$ whenever $N(b) = 1$ due to the last constraint), we obtain two possibility distributions π_1 and π_2 defined by:

$$\pi_1(\{a, b\}) = 1 \qquad \pi_1(\{b\}) = 0 \qquad \pi_1(\{a\}) = 1 \qquad \pi_1(\{\}) = 0$$

and

$$\pi_2(\{a, b\}) = 1 \qquad \pi_2(\{b\}) = 0 \qquad \pi_2(\{a\}) = 0 \qquad \pi_2(\{\}) = 0.$$

It is clear that π_2 cannot be least specific since $\pi_1 > \pi_2$. We then have that S_P^{s} only contains a single element, namely π_1. With N the necessity measure induced by π_1 we obtain $N(a) = 1$ and $N(b) = 0$. The unique answer set of P is thus $\{a^1\}$, which indeed corresponds with the answer set of the ordinary disjunctive program $P' = \{a; b \leftarrow, a \leftarrow b\}$.

4 Weak Possibilistic Semantics

The semantics we discussed thus far have a clear non-deterministic flavor: if the antecedent is known, a rule declares that we should choose one or more consequents to accept (and at the same time choose as few as possible). Under this non-deterministic view the rule '$a; b \leftarrow$' means that a is believed to be true or b is believed to be true. However, there are cases in which we do not want to or cannot make a commitment. In other words, we want a rule '$a \vee b \leftarrow$' to mean that *a or b* is true, without clarifying whether it is a, b or both that are true. In this sense we regard answer sets more as epistemic states than as possible solutions to a problem.

In this section we first define an alternative semantics for disjunctive ASP (different from the one in Section 2.1) and then extend it to possibilistic ASP. Before we give an example of the semantics, we first note that when we use weak disjunction it matters whether we model a sentence like "when it is raining or snowing, you will get wet" as either $wet \leftarrow rain \vee snow$ or as the set of rules $\{wet \leftarrow rain, wet \leftarrow snow\}$. Indeed, the latter implies that we can only derive '*wet*' after we have made the choice between '*rain*' or '*snow*'. To accommodate for this we need to slightly alter the syntax of ASP.

Definition 3. *A* clause *e is a disjunction of one or more atoms. A* clausal rule *is an expression of the form $e_0 \leftarrow e_1, ..., e_m, not\ e_{m+1}, ..., not\ e_n$ with e_i a clause for every $0 \leq i \leq n$. The clause e_0 is called the head of the rule and $e_1, ..., e_m, not\ e_{m+1}, ..., not\ e_n$ the body of the rule. A* clausal program *is a finite set of clausal rules. A* positive clausal program *is a finite set of* positive clausal rules *which are expressions of the form $e_0 \leftarrow e_1, ..., e_m$. The* Herbrand base $\mathcal{B}_P$ *of a clausal program P is redefined as being the set of clauses appearing in P.*

It is easy to see that disjunctive programs are a special case of clausal programs. We can now take a look at an example.

Example 5. We live in Ohio and we want to book a holiday trip. Either we go to Venice, Rome or Florida (USA). After we have selected our destination, we can book our trip. Also, as soon as we know that we go to either Venice or Rome we need to arrange our visa so that we have it well before our departure date. We have program P with the rules:

$$\begin{aligned} \mathit{venice} \vee \mathit{rome} \vee \mathit{florida} &\leftarrow \\ \mathit{book\_transportation} &\leftarrow \mathit{venice} \\ \mathit{book\_transportation} &\leftarrow \mathit{rome} \\ \mathit{book\_transportation} &\leftarrow \mathit{florida} \\ \mathit{arrange\_visa} &\leftarrow \mathit{venice} \vee \mathit{rome} \end{aligned}$$

Definition 4. *For a positive clausal program P we define the immediate consequence operator T_P^{w}* w.r.t. *a set of clauses E as:*

$$T_P^{\mathrm{w}}(E) = \{e_0 \mid e_0 \leftarrow e_1, ..., e_m \in P \wedge \forall i \in \{1, ..., m\} \cdot \exists e \in E \cdot e \subseteq e_i\}$$

where we identify a clause with its set of atoms. We use $P^\star_{\mathrm{w}}$ to denote the fixpoint which is obtained by repeatedly applying T_P^{w} starting from the empty interpretation, i.e. *this is the least fixpoint of T_P^{w}* w.r.t. *set inclusion. An interpretation E is called an answer set of a positive clausal program P iff it is a minimal set of clauses for which $E \models P^\star_{\mathrm{w}}$.*

Proposition 3. *The operator T_P^{w} is monotonic.*

Note that this definition of the immediate consequence operator is a generalization of the immediate consequence operator for a definite program from Section 2.1. Indeed, for a positive clausal program where all clauses contain only a single atom, *i.e.* a definite program, we have that $P^\star = P^\star_{\mathrm{w}}$. Also note that a positive clausal program always has a unique answer set.

Example 6. We again consider Example 5. It is easy to see that the unique answer set of program P is $\{\mathit{venice} \vee \mathit{rome} \vee \mathit{florida}\}$. Indeed, without any further information we cannot derive anything but the disjunction itself.

Proposition 4. *Let P be a positive clausal program. We can compute the unique answer set of P in polynomial time.*

Thus we find that weak disjunction has a lower complexity than strong disjunction (which is in NP when we only have rules without negation-as-failure and otherwise in Σ_2^{P} [1]). This is clearly due to the fact that there is no non-determinism and that the unique answer set can be found using an iterative procedure. This does, however, imply that we can reason with certain forms of disjunction in an intuitive way without requiring additional complexity. This is an advantage of weak disjunction since many situations that involve disjunction and have uncertainty tend to lead to an epistemic view, for which weak disjunction offers a lower complexity than strong disjunction.

To define the concept of an answer set of an arbitrary (not necessarily positive) clausal program we also need to redefine the reduct for clausal programs. As in the ordinary case, we want that '*not* e' is removed whenever the clause 'e' is satisfied, *i.e.* '*not* e' is true when there does not exist an $e' \in E$ such that $e' \subseteq e$. In other words, the guess E reflects what we think that we are capable of deriving from the program and we need to verify that this is indeed the case.

Definition 5. *Given a clausal program P and a set of clauses E, the reduct P^E of P* w.r.t. *E is defined as*

$$\begin{aligned} P^E = \{ & e_0 \leftarrow e_1, ..., e_m \;\mid\; \forall i \in \{m+1, ..., n\} \cdot \forall e \in E \cdot e \not\subseteq e_i \\ & \wedge (e_0 \leftarrow e_1, ..., e_m, \mathit{not}\ e_{m+1}, ..., \mathit{not}\ e_n) \in P \}. \end{aligned}$$

We say that E is an answer set of the clausal program P iff $(P^E)^\star_{\mathrm{w}} = E$, *i.e.* if E is the answer set of the reduct P^E.

Example 7. Consider the following clausal program P:

$$a \vee b \leftarrow \qquad c \leftarrow \qquad d \leftarrow \mathit{not}\ (a \vee b \vee d) \qquad e \leftarrow \mathit{not}\ c.$$

The reduct P^E with $E = \{a \vee b, c, e\}$ is then:

$$a \vee b \leftarrow \qquad c \leftarrow$$

since $\{a, b\} \subseteq \{a, b, d\}$ and $\{c\} \subseteq \{c\}$. The answer set of the reduct P^E is given by $(P^E)^\star_{\mathrm{w}} = \{a \vee b, c\}$, hence E is not an answer set of P since $(P^E)^\star_{\mathrm{w}} \neq E$.

We now extend the semantics to the case of *possibilistic clausal programs* which are finite sets of *possibilistic clausal rules* $p = (r, \lambda)$ with r a clausal rule.

Definition 6. *Let P be a possibilistic clausal program. Let $V : \mathcal{B}_P \to [0, 1]$ be a valuation. For every $p \in P$ where we have that $p = (r, \lambda)$ and $r = (e_0 \leftarrow e_1, ..., e_m, \mathit{not}\ e_{m+1}, ..., \mathit{not}\ e_n)$ the constraint γ_V^{w} induced by r under the weak possibilistic semantics is given by*

$$N(e_0) \geq \min\{N(e_1), ..., N(e_m), 1 - V(e_{m+1}), ..., 1 - V(e_n), \lambda\}. \tag{3}$$

$C^{\mathrm{w}}_{(P,V)} = \{\gamma_V^{\mathrm{w}}(r) \mid r \in P\}$ is the set of constraints imposed by program P and V. A possibility distribution that satisfies the constraints in $C^{\mathrm{w}}_{(P,V)}$ is called a possibilistic model of $C^{\mathrm{w}}_{(P,V)}$. We write $S^{\mathrm{w}}_{(P,g)}$ to denote the set of all least specific possibilistic models of $C^{\mathrm{w}}_{(P,V)}$. V is called a possibilistic answer set of P iff there exists a $\pi \in S^{\mathrm{s}}_{(P,V)}$ such that $\forall e \in \mathcal{B}_P \cdot V(e) = N(e)$.

Example 8. Let us consider the program P from Example 2, yet this time using the weak semantics for disjunction. We have the program Q with the single rule

$$\mathbf{0.7:}\ a \vee b \leftarrow .$$

The set of constraints C_Q^{w} is this time around given by $\{N(a \vee b) \geq 0.7\}$. The unique possibilistic answer set of Q is given by $\left\{(a \vee b)^{0.7}\right\}$, which is also an intuitively satisfactory result.

We now direct our attention to the discussion of how one should treat a constraint, which are rules of the form '$\leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n$', in a (disjunctive) possibilistic ASP program.[3] If we look at the intuition that underlies constraints in ordinary ASP, then it seems natural to treat a constraint such as '$\leftarrow a, b$' as $N(a \wedge b) = 0$, *i.e.* it is not possible that both atoms are true at the same time. The next definition extends Definition 6 by formalizing possibilistic constraints.

Definition 7. *Let P be a possibilistic clausal program. Let $V : \mathcal{B}_P \rightarrow [0, 1]$ be a valuation. For every* possibilistic constraint *p with $p \in P$, $p = (r, \lambda)$ and $r = (\leftarrow e_1, ..., e_m, not\ e_{m+1}, ..., not\ e_n)$ we define the associated constraint γ_g^{w} induced by r by*

$$N(e_1 \wedge ... \wedge e_m) \leq 1 - \min\{\lambda, 1 - V(e_{m+1}), ..., 1 - V(e_n)\}. \tag{4}$$

Example 9. If we once again consider Example 5 we can extend the program P with the rule '$\leftarrow$ *florida*'. The unique answer set of the program is then $\{rome \vee venice, arrange\_visa\}$. Indeed, with the additional knowledge that we will not be traveling to Florida we can readily conclude that we should arrange our visa. However, as desired, we still cannot conclude that we should book our transportation, which we can only do as soon as we pick either Venice or Rome as the actual destination.

It is important to note, however, that adding constraints to a positive clausal program affects its complexity. Indeed, finding whether an atom belongs to an answer set of a positive clausal program which, in addition, has possibilistic constraints, is NP-complete.

Proposition 5. *Let P be a possibilistic positive clausal program with possibilistic constraints. Finding whether P has a possibilistic answer set is* NP*-complete.*

Proof. (sketch) We can readily simulate 3SAT using positive clausal programs and possibilistic constraints. Let $\phi = (l_{11} \vee ... \vee l_{13}) \wedge \ldots \wedge (l_{n1} \vee ... \vee l_{n3})$ be an expression in conjunctive normal form where all clauses have 3 literals (*i.e.* either

[3] Note that one way to treat such constraints in ordinary ASP is to simulate these constraints as $\theta \leftarrow not\ \theta, a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n$ with θ a fresh atom. However, similar as to Example 3 we would end up with many possibilistic answer sets, *i.e.* the simulation does not succeed in eliminating undesired solutions.

an atom or the negation of an atom). Let P be a positive clausal program with a rule $p = (r, 1)$ and $r = (l'_{i1} \vee ... \vee l'_{i3} \leftarrow)$ for every clause in ϕ and where l' is l when $l = a$ is an atom and l' is a fresh atom $\bar{l}$ when $l = \neg a$. For every literal l in ϕ we also add the possibilistic constraint $p = (r, 1)$ with $r = (\leftarrow l, \bar{l})$. It is then easy to see that ϕ has a model if and only if P has an answer set. □

The same complexity result hold for clausal programs without certainty weights.

5 Related Work

A large body of research has been devoted to combining uncertainty with ASP. Different approaches are proposed in the literature depending on whether the uncertainty is treated in a qualitative or quantitative way. When uncertainty is treated in a quantitative way, probability theory seems to be the most often used. For example, in [2] uncertain information is encoded as a probabilistic atom which, intuitively, describes the probability that the atom will take on a certain value in some random selection given some other known evidence.

The most popular approach for dealing with uncertainty in a qualitative way is possibility theory. Combining possibility theory with logic programming was an idea first proposed in [7]. The work in [13] was one of the first papers to explore the idea of combining possibility theory with ASP. This work was later extended to also cover the case of disjunctive ASP in [14]. It was, however, noted in [3] that the semantics from [13, 14] offer unintuitive results in certain cases since neither approach takes the certainty into account when dealing with negation-as-failure. This problem was discussed in [3] and a new characterization of normal ASP programs was established based on constraints on possibility distributions, which can naturally be generalized to cover possibilistic normal ASP programs.

Alternative semantics for PASP exist in the form of pstable models [15, 4]. Yet these models are closer to the intuition of classical models than they are to the intuition of stable models as used in ASP. Hence the intuition that is captured by pstable models is different, where the focus is more on finding reasonable results in programs faced with uncertainty and which are inconsistent. There is a formal connection between the approach from [3] and the work on residuated logic programs [5] under the Gödel semantics. Both approaches are different in spirit, however, in the same way that possibilistic logic (which deals with uncertainty or priority) is different from Gödel logic (which deals with graded truth). The formal connection is due to the fact that necessity measures are min-decomposable. The work in this paper clearly differs from the work on residuated logic programs since necessity measure are not max-decomposable, which highlights that possibilistic logic is not truth-functional in general [8].

6 Conclusion

In this paper we have introduced the semantics of possibilistic disjunctive ASP in terms of constraints on possibility distributions. This provides us with natural

semantics for dealing with possibilistic disjunctive ASP pervaded by uncertainty. We explored two different views of disjunction, the non-deterministic view as found in ordinary disjunctive ASP as well as a more epistemic view of disjunction. These two views are unearthed by the two distinct ways in which we can interpret a disjunctive rule as a constraint on possibility distributions. Due to the epistemic nature of possibilistic logic we find that the epistemic view of disjunction is oftentimes the one that offers the most intuitive understanding of the problem. Finally, we also examined the complexity of weak disjunction.

References

1. Baral, C.: Knowledge, Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Baral, C., Gelfond, M., Rushton, N.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming 9(1), 57–144 (2009)
3. Bauters, K., Schockaert, S., De Cock, M., Vermeir, D.: Possibilistic answer set programming revisited. In: Proc. of UAI'10 (2010)
4. Confalonieri, R., Nieves, J.C., Vázquez-Salceda, J.: Pstable semantics for logic programs with possibilistic ordered disjunction. In: Proc. of AI*IA'09. pp. 52–61 (2009)
5. Damásio, C.V., Pereira, L.M.: Monotonic and residuated logic programs. In: Proc. of ECSQARU'01. pp. 748–759 (2001)
6. Dubois, D., Lang, J., Prade, H.: Towards possibilistic logic programming. In: Proc. of ICLP'91. pp. 581–595 (1991)
7. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. Handbook of Logic for Artificial Intelligence and Logic Programming 3(1), 439–513 (1994)
8. Dubois, D., Prade, H.: Can we enforce full compositionality in uncertainty calculi? In: Proc. of AAAI'94. pp. 149–154 (1994)
9. Dubois, D., Prade, H., Schockaert, S.: Rules and meta-rules in the framework of possibility theory and possibilistic logic. Scientia Iranica (2011), to appear
10. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
11. Gelfond, M., Lifzchitz, V.: The stable model semantics for logic programming. In: Proc. of ICLP'88. pp. 1081–1086 (1988)
12. Lifschitz, V., Schwarz, G.: Extended logic programs as autoepistemic theories. In: Proc. of LPNMR'93. pp. 101–114 (1993)
13. Nicolas, P., Garcia, L., Stéphan, I., Lefèvre, C.: Possibilistic uncertainty handling for answer set programming. Annals of Mathematics and Artificial Intelligence 47(1–2), 139–181 (2006)
14. Nieves, J.C., Osorio, M., Cortés, U.: Semantics for possibilistic disjunctive programs. In: Proc. of LPNMR'07. pp. 315–320 (2007)
15. Osorio, M., Pérez, J.A.N., Ramírez, J.R.A., Macías, V.B.: Logics with common weak completions. Journal of Logic and Computation 16(6), 867–890 (2006)
16. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5(2), 285–309 (1955)
17. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems pp. 3–28 (1978)