Introspecting Preferences in Answer Set Programming

Zhizheng Zhang
School of Computer Science and Engineering, Southeast University, Nanjing, China
seu_zzz@seu.edu.cn

Abstract
This paper develops a logic programming language, ASP\(^{EP}\), that extends answer set programming language with a new epistemic operator \(≽_x\) where \(x \in \{♯, ⊇\}\). The operator are used between two literals in rules bodies, and thus allows for the representation of introspections of preferences in the presence of multiple belief sets: \(G ≽_♯ F\) expresses that \(G\) is preferred to \(F\) by the cardinality of the sets, and \(G ≽_⊇ F\) expresses \(G\) is preferred to \(F\) by the set-theoretic inclusion. We define the semantics of ASP\(^{EP}\), explore the relation to the languages of strong introspections, and study the applications of ASP\(^{EP}\) by modeling the Monty Hall problem and the principle of majority.

2012 ACM Subject Classification Computing methodologies → Logic programming and answer set programming

Keywords and phrases Answer Set, Preference, Introspection

Digital Object Identifier 10.4230/OASIcs.ICLP.2018.3

1 Introduction

Preferences have extensively been studied in disciplines such as economy, operations research, psychology, philosophy, and artificial intelligence as showed in [8], [18], [2], [15], and [7] etc. In [25], von Wright defined preference as a relation between states of affairs. In formal logical languages, states of affairs are typically represented as propositions. Follow this tradition, one of the important directions in artificial intelligence is the logical representation and reasoning of preferences. Many extensions of the languages of answer set programming (ASP) have been developed for handling preferences due to the strong power of ASP in expressing defaults. Those languages provide elegant methodologies for modeling the intractable problems with defaults and preferences. Examples include the ordered logic programming [20], the logic programming with ordered disjunction [4], the answer set optimization [5][3], the prioritized logic programming [19], the CR-prolog [1], the possibilistic answer set programming [17] etc. The preferences handled in those answer set programs are used to evaluate the preferred answer sets via specifying the precedence over the rules or the literals in rules heads.

Different from the above answer set programming paradigms with preferences, our purpose in this paper is to represent introspections of preferences over propositions in the presence of multiple belief sets by proposing a new epistemic operator \(≽_x\) where \(x \in \{♯, ⊇\}\). For propositions \(F\) and \(G\), \(F ≽_♯ G\) expresses that \(F\) is true in more belief sets than \(G\), and can be read as “\(F\) is more possible than \(G\)”. And \(F ≽_⊇ G\) expresses that \(F\) is always true in the belief sets where \(G\) is true, which tells “\(F\) is antecedent to \(G\)” or “\(F\) is true whenever \(G\) is true” etc. We first demonstrate this motivation using an example from our family life.

---

\(^1\) This work is supported by the National Key Research and Development Plan of China (No. 2017YFB1002801).
Table 1 Combo of Attractions.

(a) Packages Information

<table>
<thead>
<tr>
<th>Package</th>
<th>Attractions</th>
<th>Ages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a₁</td>
<td>Kids, teens</td>
</tr>
<tr>
<td></td>
<td>a₂</td>
<td>Adults</td>
</tr>
<tr>
<td></td>
<td>a₃</td>
<td>teens</td>
</tr>
<tr>
<td>2</td>
<td>b₁</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>b₂</td>
<td>Adults</td>
</tr>
<tr>
<td></td>
<td>b₃</td>
<td>Kids</td>
</tr>
<tr>
<td>3</td>
<td>c₁</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>c₂</td>
<td>teens</td>
</tr>
<tr>
<td></td>
<td>c₃</td>
<td>Kids</td>
</tr>
</tbody>
</table>

(b) Possible Combinations of Attractions

<table>
<thead>
<tr>
<th>Package</th>
<th>Combinations</th>
<th>Age Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a₁,a₂}</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>{a₁,a₃}</td>
<td>Kids, teens</td>
</tr>
<tr>
<td></td>
<td>{a₂,a₃}</td>
<td>Adults, teens</td>
</tr>
<tr>
<td>2</td>
<td>{b₁,b₂}</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>{b₁,b₃}</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>{b₂,b₃}</td>
<td>Adults, Kids</td>
</tr>
<tr>
<td>3</td>
<td>{c₁,c₂}</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>{c₁,c₃}</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>{c₂,c₃}</td>
<td>teens, Kids</td>
</tr>
</tbody>
</table>

Example 1. Consider three discount packages offered by an amusement resort as showed in Table 1(a). Each of them contains three attractions but only two of them are available. A family is allowed to buy at most one package in advance, and may determine which two attractions to choose according to the actual situations, such as the waiting time, physical situation, when they are in the resort. For instance, a family with a kid child and a teenage boy decide which package to buy by the following criteria: (1) The family prefer the package that promises more opportunities for the kid child; (2) The parents request that their teenage boy has an attraction to visit whenever they visit an attraction.

Directly, the packages information allow the family to have nine possible combinations of attractions as showed in the table 1(b).

And the family can have the following three conclusions via simple counting.

(i) Both package 2 and package 3 provide more opportunities for the kid child than package 1.

(ii) Both package 1 and package 3 guarantee that the teenage boy has an attraction to visit whenever the parents visit an attraction.

(iii) By (i) and (ii), Package 3 should be the favorite package for the family.

It is easy to get the combinations by encoding the packages information and the purchase requirements in a logic program $\Pi_{ep}$ containing the following rules:

---

2 In the tables, “All” means that there is no age limitation.

3 To avoid the boy running around without parents.
that has exactly nine answer sets which correspond to the nine possible combinations in Table 1. We now expect to expand $\Pi_{ep}$ by rules that is able to intuitively represent the criteria such that the result program is able to give the conclusions as showed in (i),(ii), and (iii). It is easy to see, for achieving the above goal, our representation and reasoning system should have an introspective ability that is able to look at the preferences over the beliefs with regard to those belief sets/answer sets.

Specifically, this paper will address the issue of introspection of preferences illustrated in the above example. We develop a logic programming language, $\text{ASP}^{\text{EP}}$, that extends the answer set programming language with a new epistemic operator $\succeq_x$ where $x \in \{\#, \supset\}$. In $\text{ASP}^{\text{EP}}$, the operator is used between two literals in rules bodies, and thus allows for the representation of introspections of preferences. Consider rules $r_3$:

\begin{verbatim}
prefer(X,Y,kid) ← age_interest(X,kids) \geq_1 age_interest(Y,kids),
package(X), package(Y)
and $r_5^-$:
request(X) ← age_interest(X,teens) \geq_2 age_interest(X,adults), package(X)
\end{verbatim}

They are able to represent the criteria (1) and (2) in the motivation example respectively.

The rest of the paper is organized as follows. In the next section, we review the basic principles underlying the answer set semantics of logic programs. In section 3, we introduce syntax and semantics of $\text{ASP}^{\text{EP}}$. In section 4, we consider the relationship between $\text{ASP}^{\text{EP}}$ and the strong introspection specification languages. In section 6, we explore the applications of $\text{ASP}^{\text{EP}}$. We conclude in section 7 with some further discussion.
2 Answer Set Programming

Throughout this paper, we assume a finite first-order signature $\sigma$ that contains no function constants of positive arity. There are finitely many Herbrand interpretations of $\sigma$, each of which is finite as well. We follow the description of ASP from [14]. A logic program over $\sigma$ is a collection of rules of the form

$$l_1 \lor \ldots \lor l_k \leftarrow l_{k+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where the $l$s are literals of $\sigma$, not is called negation as failure, or is epistemic disjunction. The left-hand side of a rule is called the head and the right-hand side is called the body. A rule is called a fact if its body is empty and its head contains only one literal, and a rule is called a denial if its head is empty. A logic program is called ground if it contains no variables. [14] intuitively interprets that an answer set associated with a ground logic program is a set of beliefs (collection of ground literals) and is formed by a reasoner guided by three principles:

- Rule’s Satisfiability principle: Believe in the head of a rule if you believe in its body.
- Consistency principle: Do not believe in contradictions.
- Rationality Principle: Believe nothing you are not forced to believe.

The definition of the answer set is extended to any non-ground program by identifying it with the ground program obtained by replacing every variable with every ground term of $\sigma$. It is worthy noting that $\top$ can be removed if it is in the body of a rule, the rule can be removed from the program if $\bot$ is in its body.

3 The ASP$^{\text{EP}}$ Language

3.1 Syntax

An ASP$^{\text{EP}}$ program $\Pi$ is a set of rules of the form

$$l_1 \lor \ldots \lor l_k \leftarrow e_1, \ldots, e_m, s_1, \ldots, s_n.$$  

where $k \geq 0$, $m \geq 0$, $n \geq 0$, the $l$s are literals in first order logic language and are called objective literals here, $e$s are extended literals which are 0-place connectives $\top$ and $\bot$, or objective literals possibly preceded by not, $s$s are subjective literals of the form $e \succ_x e'$ or $e \nsucc_x e'$ where $e$ and $e'$ are extended literals and $x \in \{\# , \supset\}$. The left-hand side of a rule is called the head and the right-hand side is called the body. As in usual logic programming, a rule is called a fact if its body is empty and its head contains only one literal, and a rule is called a denial if its head is empty. We use head($r$) to denote the set of objective literals in the head of a rule $r$ and body($r$) to denote the set of extended literals and subjective literals in the body of $r$. Sometimes, we use head($r$) $\leftarrow$ body($r$) to denote a rule $r$. The positive body of a rule $r$ is composed of the extended literals containing no not in its body. We use body$^+(r)$ to denote the positive body of $r$. $r$ is said to be safe if each variable in it appears in the positive body of the rule. We will use sl(II) to denote the set of subjective literals appearing in $\Pi$.

It is clear that an ASP$^{\text{EP}}$ program containing no subjective literals is a disjunctive logic program that can be dealt with by ASP solvers like DLV [9], CLASP [10].

It is worthy of noting that, for convenient description, we will use $e \succ_x e'$ to denote the strict preference that can be expressed by the conjunction of $e \succ_x e'$ and $e' \npreceq_x e$, and use $e \approx_x e'$ to denote the preferential indifference that can be expressed by the conjunction of
We will restrict our definition of the semantics to ground programs. However, we admit rule schemata containing variables bearing in mind that these schemata are just convenient representations for the set of their ground instances. In the following definitions, \( l \) is used to denote a ground objective literal, \( e \) is used to denote a ground extended literal, and \( s \) is used to denote a ground subjective literal.

### 3.2 Semantics

We will restrict our definition of the semantics to ground programs. However, we admit rule schemata containing variables bearing in mind that these schemata are just convenient representations for the set of their ground instances. In the following definitions, \( l \) is used to denote a ground objective literal, \( e \) is used to denote a ground extended literal, and \( s \) is used to denote a ground subjective literal.

#### 3.2.1 Satisfiability

Let \( W \) be a non-empty collection of consistent sets of ground objective literals, \((W, w)\) is a pointed ASP\(^E\) structure of \( W \) where \( w \in W \). \( W \) is a model of a program \( \Pi \) if for each rule \( r \) in \( \Pi \), \( r \) is satisfied by every pointed ASP\(^E\) structure of \( W \). The notion of satisfiability denoted by \( \models_{ep} \) is defined below.

- \((W, w)\models_{ep} \top\)
- \((W, w)\not\models_{ep} \bot\)
- \((W, w)\models_{ep} l \) if \( l \in w \)
- \((W, w)\not\models_{ep} l \) if \( l \not\in w \)
- \((W, w)\models_{ep} e \succ e' \) if \(|\{w \in W : (W, w)\models_{ep} e\}| \geq |\{v \in W : (W, v)\models_{ep} e'\}|\)
- \((W, w)\models_{ep} e \succ e' \) if \(|\{w \in W : (W, w)\models_{ep} e\}| \geq \{v \in W : (W, v)\models_{ep} e'\}\)
- \((W, w)\models_{ep} e \not\succ e' \) if \((W, w)\not\models_{ep} e \succ e', x \in \{\succ, \approx\}\)

Then, for a rule \( r \) in \( \Pi \), \((W, w)\models_{ep} r \) if

- \( \exists l \in head(r) : (W, w)\models_{ep} l \), or
- \( \exists l \in body(r) : (W, w)\not\models_{ep} l \).

The satisfiability of a subjective literal does not depend on a specific belief set \( w \) in \( W \), hence we can simply write \( W \models_{ep} s \) if \((W, w)\models_{ep} s \) and say the subjective literal \( s \) is satisfied by \( W \), and we can simply write \( W \not\models_{ep} s \) if \((W, w)\not\models_{ep} s \) and say the subjective literal \( s \) is not satisfied by \( W \).

We consider the properties of the above satisfiability by some axioms of the strict preference relation proposed by von Wright in [25]. Let \( W \) be a non-empty collection of consistent sets of ground objective literals, the following properties of the satisfiability \( \models_{ep} \) hold.

- \( \succ_x \) Asymmetry. \( W\models_{ep} e \succ_x e' \implies W\models_{ep} e' \not\succ_x e \)
- \( \succ_x \) Inescapability. \( W\models_{ep} e \succ_x e', W\models_{ep} e'' \not\succ_x e' \implies W\models_{ep} e \succ_x e'' \)
- \( \succ_x \) Transitivity. \( W\models_{ep} e \succ_x e', W\models_{ep} e' \succ_x e'' \implies W\models_{ep} e \succ_x e'' \)
- \( \simeq_x \) Irreflexivity. \( W\models_{ep} e \not\simeq_x e \)
- \( \simeq_x \) Symmetry. \( W\models_{ep} e \simeq_x e' \implies W\models_{ep} e' \simeq_x e \)
- \( \simeq_x \) Transitivity. \( W\models_{ep} e \simeq_x e', W\models_{ep} e' \simeq_x e'' \implies W\models_{ep} e \simeq_x e'' \)
- \( \succ \) R-Analogy. \( W\models_{ep} e \succ e', W\models_{ep} e' \succ e'' \implies W\models_{ep} e \succ e'' \)
- \( \succ \) L-Analogy. \( W\models_{ep} e \succ e', W\models_{ep} e' \succ e'' \implies W\models_{ep} e' \succ e'' \)

where \( x \in \{\succ, \approx\}\).

In addition, let \( W \) be a non-empty collection of consistent sets of ground objective literals, it is easy to find that

- \( W\models_{ep} e \succ e \)
- \( W\models_{ep} \top \succ e \)
We use $\mathcal{W} \models_{ep} e \not\triangleright x \bot$

$\mathcal{W} \models_{ep} e \not\triangleright e^{not}$

where $e^{not}$ is $l$ if $e$ is not $l$, and $e^{not}$ is not $l$ if $e$ is $l$, and $\top^{not}$ is $\bot$, and $\bot^{not}$ is $\top$.

### 3.2.2 World Views

We first give the definition of candidate world view for disjunctive logic programs and arbitrary ASP$^{\text{EP}}$ programs respectively. Then, we define world view for ASP$^{\text{EP}}$ programs by presenting a minimizing preferences principle.

**Definition 2.** Let $\Pi$ be a disjunctive logic program, the candidate world view of $\Pi$ is the non-empty set of all its answer sets, written as $\text{AS}(\Pi)$.

**Definition 3.** Let $\Pi$ be an arbitrary ASP$^{\text{EP}}$ program, and $\mathcal{W}$ is a non-empty collection of consistent sets of ground objective literals in the language of $\Pi$, we use $\Pi^W$ to denote the disjunctive logic program obtained by removing the epistemic operators using the following reduct laws

1. removing from $\Pi$ all rules containing subjective literals not satisfied by $\mathcal{W}$.
2. removing all other occurrences of subjective literals of the form $e \not\triangleright x e$ or $\bot \not\triangleright x e$ or $e \not\triangleright x \bot$ or $e \not\triangleright e^{not}$.
3. replacing all other occurrences of subjective literals of the form $e \not\triangleright x \top$ by $e$.
4. replacing all other occurrences of subjective literals of the form $\bot \not\triangleright x e$ by $e^{not}$.
5. replacing other occurrences of subjective literals of the form $e_1 \triangleright x e_2$ or $e_1 \not\triangleright x e_2$ by four conjunctions $e_1, e_2$, and $e_1^{not}, e_2$, and $e_1, e_2^{not}$, and $e_1^{not}, e_2^{not}$ respectively, where $e^{not}$ is $l$ if $e$ is not $l$, and $e^{not}$ is not $l$ if $e$ is $l$, and $\top^{not}$ is $\bot$, and $\bot^{not}$ is $\top$. Then, $\mathcal{W}$ is a candidate world view of $\Pi$ if $\mathcal{W}$ is a candidate world view of $\Pi^W$.

We use $\text{cuv}(\Pi)$ to denote the set of candidate world views of an ASP$^{\text{EP}}$ program $\Pi$. $\Pi^W$ is said to be the reduct of $\Pi$ with respect to $\mathcal{W}$. Such a reduct process eliminates subjective literals so that the belief sets in the model are identified with the answer sets of the program obtained by the reduct process. The intuitive meanings of the reduct laws can be described as follows:

- The first reduct law directly comes from the notion of Rule Satisfiability and Rationality Principle in answer set programming which means if a rule’s body cannot be satisfied (believed in), the rule will contribute nothing;
- The second reduct law stems from the fact $e \not\triangleright x e$ and $\bot \not\triangleright x e$ and $e \not\triangleright x \bot$ and $e \not\triangleright e^{not}$ are tautologies.
- The third reduct law states that, you are forced to believe $e$ with regard to each belief set due to the fact that $e \not\triangleright x \top$ implies $e$ is true with regard to each answer set and the Rationality Principle in ASP.
- The fourth law states that, you are forced to believe $e^{not}$ with regard to each belief set due to the fact that $\bot \not\triangleright x e$ implies $e$ is not true with regard to each answer set.
- The last law states that, both the literals $e_1$ and $e_2$ in $e_1 \not\triangleright x e_2$ may be true or not with regard to each belief set.

**Definition 4.** Let $\Pi$ be an arbitrary ASP$^{\text{EP}}$ program, and $\mathcal{W}$ is a non-empty collection of consistent sets of ground objective literals in the language of $\Pi$, $\mathcal{W}$ is a world view of $\Pi$ if it satisfies the conditions below

$\mathcal{W} \in \text{cuv}(\Pi)$

Minimizing preferences principle: $\exists \mathcal{W} \in \text{cuv}(\Pi)(\{\bar{s} | s \in \text{sl}(\Pi) \land \mathcal{W} \models_{ep} \bar{s}\} \supset \{\bar{s} | s \in \text{sl}(\Pi) \land \mathcal{W} \models_{ep} \bar{s}\})$
where $\bar{s}$ is $e \geq_x e'$ if $s$ is $e \geq_x e'$, and $\bar{s}$ is $e \not{\geq}_x e'$ if $s$ is $e \not{\geq}_x e'$.

We use $wv(\Pi)$ to denote the set of world views of an ASP$^{\mathsf{EP}}$ program $\Pi$.

**Definition 5.** Let $\Pi$ be an ASP$^{\mathsf{EP}}$ program, a ground objective literal $l$ is true in $\Pi$ (written by $\Pi \vdash_{ep} l$) if $\forall W \in wv(\Pi) \forall w \in W((W, w) \models_{ep} l)$.

**Example 6.** Consider $\Pi = \Pi_{ep} \cup \{r_3, r_2\}$ where $\Pi_{ep}$ and $r_1$ and $r_2$ are given in section 1. It is easy to see that $\Pi$ has an unique world view containing nine belief sets:

1. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(1),\text{age_interest}(1,\text{kids}),\text{age_interest}(1,\text{adults})\}$
2. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(1),\text{age_interest}(1,\text{teens}),\text{age_interest}(1,\text{adults})\}$
3. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(1),\text{age_interest}(1,\text{adults}),\text{age_interest}(1,\text{teens})\}$
4. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(2),\text{age_interest}(2,\text{kids}),\text{age_interest}(2,\text{adults}),\text{age_interest}(2,\text{teens})\}$
5. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(2),\text{age_interest}(2,\text{adults}),\text{age_interest}(2,\text{kids}),\text{age_interest}(2,\text{teens})\}$
6. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(2),\text{age_interest}(2,\text{adults}),\text{age_interest}(2,\text{kids})\}$
7. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(2),\text{age_interest}(3,\text{kids}),\text{age_interest}(3,\text{adults}),\text{age_interest}(3,\text{teens})\}$
8. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(2),\text{age_interest}(3,\text{adults}),\text{age_interest}(3,\text{kids}),\text{age_interest}(3,\text{teens})\}$
9. $\{\text{prefer}(2,1),\text{prefer}(3,1),\text{request}(1),\text{request}(3),\text{package}(3),\text{age_interest}(3,\text{teens}),\text{age_interest}(1,\text{kids})\}$

Then we have $\Pi \vdash_{ep} \text{prefer}(2,1)$ and $\Pi \vdash_{ep} \text{prefer}(3,1)$ corresponding to the conclusion (i), and $\Pi \vdash_{ep} \text{request}(3)$ and $\Pi \vdash_{ep} \text{request}(1)$ corresponding to the conclusion (ii), and it is easy to verify that if we add to $\Pi$ another rule:

$\text{buy}(X) \leftarrow \text{request}(X), \text{not \ prefer}(Y, X), \text{package}(X), \text{package}(Y), X! = Y$

that states a simple ordered-based choice strategy, then we can get $\Pi \vdash_{ep} \text{buy}(3)$ corresponding to the conclusion (iii) in section 1.

## 4 Relation to Strong Introspection Specifications

Several languages have been developed by extending the languages of answer set programming (ASP) using epistemic operators to handle introspections. The need for such extension of ASP was early recognized and addressed by Gelfond in [11], where Gelfond proposed an extension of ASP with two modal operators $K$ and $M$ and their negations (ASP$^{\mathsf{KM}}$). Informally, $K p$ expresses “$p$ is known” ($p$ is true in all belief sets of the agent), $M p$ means “$p$ may be true” ($p$ is true in some belief sets of the agent). It has been proved that ASP$^{\mathsf{KM}}$ is potential in dealing with some important issues in the field of knowledge representation and reasoning, for instance the correct representation of incomplete information in the presence of multiple belief sets [12], commonsense reasoning [12], formalization for conformant planning [16], and meta-reasoning [24] etc. Recently, there is increasing research in this direction to address the long-standing problems of unintended world views due to recursion through modalities...
that were introduced by Gelfond [11], e.g. [13, 16, 6]. Very recently, Shen and Eiter [22]
introduced general logic programs possible containing epistemic negation NOT (ASP\textsuperscript{NOT}),
and defined its world views by minimizing the knowledge. ASP\textsuperscript{NOT} can not only express K
p and M p formulas by not NOT p and NOT not p, but also offer a solution to the problems of
unintended world views. In this section we show that ASP\textsuperscript{KM} logic programs in [16] where
the most recent version of ASP\textsuperscript{KM} is defined, and a special kind of ASP\textsuperscript{NOT} programs can
be viewed as ASP\textsuperscript{EP} programs.

4.1 Relation to ASP\textsuperscript{KM}

An ASP\textsuperscript{KM} program is a set of rules of the form \( h_1 \lor \ldots \lor h_k \leftarrow b_1, \ldots, b_m \)
where \( k \geq 0 \), \( m \geq 0 \), \( h_i \) is an objective literal, and \( b_i \) is an objective literal possible preceded by a negation
as failure operator not, a modal operator K or M, or a combination operator not K or not M. For distinguishment, we call the world view of the ASP\textsuperscript{KM} program \( \text{KM-world view} \).

Let \( W \) be a non-empty collection of consistent sets of ground objective literals, \( W \) is
a candidate world view of \( \Pi \) iff \( W \) is a KM-world view of \( \Omega \) where \( \Pi_W \) is a disjunctive
logic program obtained using Modal Reduct as showed in Table 2.

In ASP\textsuperscript{KM}, the notion of satisfiability is defined from \( \models \) relationship below.

\[
\begin{align*}
&< W, w > \models_{\text{km}} l \text{ if } l \in w \\
&< W, w > \models_{\text{km}} \text{not } l \text{ if } l \notin w \\
&< W, w > \models_{\text{km}} Kl \text{ if } \forall v \in W: l \in v \\
&< W, w > \models_{\text{km}} \text{not } Kl \text{ if } \exists v \in W: l \notin v \\
&< W, w > \models_{\text{km}} Ml \text{ if } \exists v \in W: l \in v \\
&< W, w > \models_{\text{km}} \text{not } Ml \text{ if } \forall v \in W: l \notin v
\end{align*}
\]

\textbf{Definition 7.} Given an ASP\textsuperscript{KM} program \( \Omega \), an ASP\textsuperscript{EP} program is called a KM-EP-Image
of \( \Omega \), denoted by \( \text{KM-EP-Image}(\Omega) \), if it is obtained by

\begin{itemize}
  \itemReplacing all occurrences of literals of the form \( Kl \) in \( \Pi \) by \( l \succeq \top \).
  \itemReplacing all occurrences of literals of the form \( Ml \) in \( \Pi \) by \( l \notin \top \) and \( l \notin \top \) respectively.
  \itemReplacing all occurrences of literals of the form \( \text{not } Kl \) in \( \Pi \) by \( l \succeq \top \) and \( l \succeq \top \) respectively.
  \itemReplacing all occurrences of literals of the form \( \text{not } Ml \) in \( \Pi \) by \( l \notin \top \) and \( l \notin \top \).
\end{itemize}

\textbf{Theorem 8.} Let \( \Omega \) be an ASP\textsuperscript{KM} program, and \( \Pi \) be the ES-EP-Image of \( \Omega \), and \( W \) be a
non-empty collection of consistent sets of ground objective literals, \( W \) is a candidate world
view of \( \Pi \) iff \( W \) is a KM-world view of \( \Omega \).

\textsuperscript{4} Here, we view \( \text{not not } l \) as a representation of \( \text{not } l' \) where we have \( l' \leftarrow \text{not } l \) and \( l' \) is a fresh literal. It
is worthwhile to note that CLINGO is able to deal with \( \text{not not } l \).
Example 9. Consider an ASP<sub>KM</sub> program Ω:  
\[ p \leftarrow M p, \quad p \leftarrow \text{not not } p \]

Ω has an unique KM-world view \{ \{ p \} \}. Its ES-EP-Image Π contains two rules

\[ p \leftarrow \text{not } p \] \quad \[ p \leftarrow \text{not not } p \]

Then, the reduct Π\{(p)\} contains five rules

\[ p \leftarrow p, \quad p \leftarrow \text{not p}, \quad p \leftarrow \bot, \quad p \leftarrow \text{not not } p \]

which has only one answer set \{ p \}. While the reduct Π\{\{\}\} contains only one rule \( p \leftarrow \text{not not } p \) which has two answer sets \{\} and \{ p \}. Then, \{\{ p \}\} is the unique candidate world view of Π.

4.2 Relation to ASP<sub>NOT</sub>

Here, we consider the ASP<sub>NOT</sub> program that is a set of the rules of the form

\[ l_1 \lor ... \lor l_k \leftarrow e_1, ..., e_m, s_1, ..., s_n \quad \text{where } k \geq 0, \quad m \geq 0, \quad n \geq 0, \quad l_i \text{ is an objective literal, } \quad e_i \text{ is an extended literal, } \quad s_i \text{ is a subjective literal of the form NOT } e \text{ or not NOT } e. \]

For distinguishment, we call the world view of an ASP<sub>NOT</sub> program NOT-world view. Let \( W \) be a non-empty collection of consistent sets of ground objective literals, \( W \) is a candidate NOT-world view of an ASP<sub>NOT</sub> program Π if \( W = \text{AS}(Π_W) \) where \( Π_W \) is a general logic program obtained using Epistemic Reduct by (1) replacing every NOT \( F \) that is satisfied by \( W \) with \( \top \), and (2) replacing every NOT \( F \) that is not satisfied by \( W \) with \( \text{not } F \). In ASP<sub>NOT</sub>, the notion of satisfiability of a subjective formula \( \text{NOT } F \) is defined from

\[ \langle W, w \rangle \models \text{NOT } F \text{ if } \exists v \in W : v \not\models_{\text{GLP}} F \]

where the satisfaction denoted by \( \models_{\text{GLP}} \) is as the satisfaction of a formula defined in general logic programming introduced in [23]. \( W \) is a NOT-world view of an ASP<sub>NOT</sub> program Π if it is a candidate NOT-world view satisfying maximal set of literals of the form NOT \( e \) appearing in Π.

Definition 10. Given an ASP<sub>NOT</sub> program Ω, an ASP<sub>EP</sub> program is called a NOT-EP-Image of Ω, denoted by NOT-EP-I(Ω), if it is obtained by

- Replacing all occurrences of literals of the form \( \text{not NOT } e \) in Ω by \( e \not\geq \top \).
- Replacing all occurrences of literals of the form NOT \( e \) in Ω by \( e \geq \top \) and \( \text{not } e \) respectively.

Theorem 11. Let Ω be an ASP<sub>NOT</sub> program, and Π be the NOT-EP-Image of Ω, and \( W \) be a non-empty collection of consistent sets of ground objective literals, \( W \) is a world view of Π iff \( W \) is a NOT-world view of Ω.

Example 12. Consider an ASP<sub>NOT</sub> program from [22] that contains two rules

\[ \text{innocent}(john) \mid \text{guilty}(john) \quad \text{innocent}(john) \leftarrow \text{not guilty}(john) \]

Ω has an unique NOT-world view \{ \{\text{innocent}(john)\}\}. The NOT-EP-Image of Ω has three rules

\[ \text{innocent}(john) \mid \text{guilty}(john) \]
\[ \text{innocent}(john) \leftarrow \text{guilty}(john) \not\geq \top \]
\[ \text{innocent}(john) \leftarrow \text{not guilty}(john) \]

and a unique world view \{ \{\text{innocent}(john)\}\}. 
5 Applications

Consider the relationship between ASP\textsuperscript{EP} and the languages of strong introspections mentioned in section 5. ASP\textsuperscript{EP} is potential in dealing with some important issues. In this section, we illustrate the use of ASP\textsuperscript{EP} in modeling problems with introspective preferences.

5.1 Describing the Principle of Majority

The principle of majority (PM) is a widely used epistemic commonsense in the fields of information fusion, decision making, social choice, etc, where incomplete information usually causes multiple belief sets, and queries are usually answered by the principle of majority. For example, consider the behavior of common birds modeled by a program PM as below:

\[
\begin{align*}
pigeon(X) &\text{ or } raven(X) &\text{ or } swallow(X) &\text{ or } sparrow(X) &\leftarrow &\text{ commonBird}(X) \\
behavior(X, migratory) &\leftarrow &swallow(X) \\
behavior(X, resident) &\leftarrow &pigeon(X) \\
behavior(X, resident) &\leftarrow &raven(X) \\
behavior(X, resident) &\leftarrow &sparrow(X)
\end{align*}
\]

Then, given a fact \( f \):

\[
\text{commonBird}(tom)
\]

and answer the query \( \text{behavior}(tom, ?) \) by the principle of majority described by the following rules \( r_r, r_m, \) and \( r_u \):

\[
\begin{align*}
behavior(X, resident) &\leftarrow behavior(X, resident) \succ_2 behavior(X, migratory), bird(X) \\
behavior(X, migratory) &\leftarrow behavior(X, migratory) \succ_1 behavior(X, resident), bird(X) \\
behavior(X, unknown) &\leftarrow behavior(X, migratory) \approx_2 behavior(X, resident), bird(X)
\end{align*}
\]

They express that a bird \( X \) is a resident(migratory) bird if \( X \) being resident(migratory) is strictly more possible than \( X \) being migratory(resident), otherwise it is unknown. It is easy to see that the program \( PM \cup \{ f, r_r, r_m, r_u \} \) gives answer \( \text{behavior}(tom, resident) \) to the query, that is

\[
PM \cup \{ f, r_r, r_m, r_u \} \models_{\text{EP}} \text{behavior}(tom, resident)
\]

5.2 Modeling the Monty Hall Problem

We will use ASP\textsuperscript{EP} to solve the Monty Hall problem from [21]: One of the three boxes labeled 1, 2, and 3 contains the keys to that new 1975 Lincoln Continental. The other two are empty. If you choose the box containing the keys, you win the car. A contestant is asked to select one of three boxes. Once the player has made a selection, Monty is obligated to open one of the remaining boxes which does not contain the key. The contestant is then asked if he would like to switch his selection to the other unopened box, or stay with his original choice. Here is the problem:does it matters if the contentant switches? The answer is YES.

One of many solutions of the Monty Hall Problem is by arithmetic [21], where nine possible states are given as showed in Table 3, and the idea in the solution can be described naturally as: Contestant switches if SWITCH can bring more wins than STAY, Contestant stays if STAY can bring more wins than SWITCH.
Table 3 Possible Results of MHP.

<table>
<thead>
<tr>
<th>Keys are in box</th>
<th>Contestant choose box</th>
<th>Monty can open box</th>
<th>Contestant switches</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2 or 3</td>
<td>2 or 3</td>
<td>loses</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>wins</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>wins</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>wins</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1 or 3</td>
<td>1 or 3</td>
<td>loses</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>wins</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>wins</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>wins</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1 or 2</td>
<td>1 or 2</td>
<td>loses</td>
</tr>
</tbody>
</table>

Encode the definition of the problem using a disjunctive logic program $MHP$ below.

\begin{align*}
\text{box}(1) \\
\text{box}(2) \\
\text{box}(3) \\
1\{\text{choose\_box}(X) : \text{box}(X)\}1 \\
1\{\text{key\_in\_box}(X) : \text{box}(X)\}1 \\
\text{can\_open\_box}(X) \leftarrow \text{box}(X), \neg \text{choose\_box}(X), \neg \text{key\_in\_box}(X) \\
\text{win\_by\_switch} \leftarrow \text{choose\_box}(X), \neg \text{key\_in\_box}(X) \\
\text{win\_by\_stay} \leftarrow \text{choose\_box}(X), \text{key\_in\_box}(X)
\end{align*}

Represent the idea in the solution by two rules $r_1$:

\[ \text{switch} \leftarrow \text{win\_by\_switch} \models \text{win\_by\_stay}, \text{win\_by\_stay} \not\models \text{win\_by\_switch} \]

and $r_2$:

\[ \text{stay} \leftarrow \text{win\_by\_stay} \models \text{win\_by\_switch}, \text{win\_by\_switch} \not\models \text{win\_by\_stay} \]

Then, we have the following result that gives a correct answer for the problem.

**Theorem 13.** $MHP \cup \{r_1, r_2\} \models_{ep} \text{switch}$ and $MHP \cup \{r_1, r_2\} \not\models_{ep} \text{stay}.$

## 6 Conclusion and Future Work

We present a logic programming formalism capable of reasoning that combines nonmonotonic reasoning, epistemic preferential reasoning, which is built on the existing efficient answer set solvers. This makes it an elegant way to formalize some problems with defaults and introspections of preferences.

A limitation of the work in this paper is that we do not consider the relationships between ASP$^{EP}$ and other well developed formalisms of preferences.

As a next goal, we will consider the introspection of other types of preferences which are considered in the AI field [8, 18]. Our future work also includes the mathematical properties of ASP$^{EP}$ programs, the methodologies for modeling with ASP$^{EP}$, and the efficient solver of ASP$^{EP}$ programs.
References


