Introspecting Preferences in Answer Set Programming

Zhizheng Zhang[1]
School of Computer Science and Engineering, Southeast University, Nanjing, China
seu_zzz@seu.edu.cn

Abstract

This paper develops a logic programming language, ASP$^{\mathrm{EP}}$, that extends answer set programming language with a new epistemic operator $\succcurlyeq_x$ where $x \in \{\sharp, \supseteq\}$. 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: $G \succcurlyeq_\sharp F$ expresses that G is preferred to F by the cardinality of the sets, and $G \succcurlyeq_\supseteq F$ expresses G is preferred to F by the set-theoretic inclusion. We define the semantics of ASP$^{\mathrm{EP}}$, explore the relation to the languages of strong introspections, and study the applications of ASP$^{\mathrm{EP}}$ 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 $\succcurlyeq_x$ where $x \in \{\sharp, \supseteq\}$. For propositions F and G, $F \succcurlyeq_\sharp G$ expresses that F is true in more belief sets than G, and can be read as *"F is more possible than G"*. And $F \succcurlyeq_\supseteq G$ 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

Package	Attractions	Ages
1	a_1	Kids,teens
	a_2	Adults
	a_3	teens
2	b_1	All
	b_2	Adults
	b_3	Kids
3	c_1	All
	c_2	teens
	c_3	Kids

(b) Possible Combinations of Attractions]

Package	Combinations	Age Interest
1	$\{a_1,a_2\}$	All
	$\{a_1,a_3\}$	Kids,teens
	$\{a_2,a_3\}$	Adults,teens
2	$\{b_1,b_2\}$	All
	$\{b_1,b_3\}$	All
	$\{b_2,b_3\}$	Adults,Kids
3	$\{c_1,c_2\}$	All
	$\{c_1,c_3\}$	All
	$\{c_2,c_3\}$	teens,Kids

▶ **Example 1.** Consider three discount packages offered by an amusement resort as showed in Table 1(a)[2]. 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*[3].

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 Π_{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.

$$1\{package(1); package(2); package(3)\}1$$
$$2\{attraction(a_1); attraction(a_2); attraction(a_3)\}2 \leftarrow package(1)$$
$$2\{attraction(b_1); attraction(b_2); attraction(b_3)\}2 \leftarrow package(2)$$
$$2\{attraction(c_1); attraction(c_2); attraction(c_3)\}2 \leftarrow package(3)$$
$$age(kids) \leftarrow attraction(a_1)$$
$$age(adults) \leftarrow attraction(a_2)$$
$$age(teens) \leftarrow attraction(a_3)$$
$$age(all) \leftarrow attraction(b_1)$$
$$age(adults) \leftarrow attraction(b_2)$$
$$age(kids) \leftarrow attraction(b_3)$$
$$age(all) \leftarrow attraction(c_1)$$
$$age(teens) \leftarrow attraction(c_2)$$
$$age(kids) \leftarrow attraction(c_3)$$
$$age(kids) \leftarrow age(all)$$
$$age(teens) \leftarrow age(all)$$
$$age(adults) \leftarrow age(all)$$
$$age\_interest(X, Y) \leftarrow package(X), age(Y).$$

that has exactly nine answer sets which correspond to the nine possible combinations in Table 1. We now expect to expand Π_{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, ASP$^{\mathrm{EP}}$, that extends the answer set programming language with a new epistemic operator $\succcurlyeq_x$ where $x \in \{\sharp, \supseteq\}$. In ASP$^{\mathrm{EP}}$, the operator is used between two literals in rules bodies, and thus allows for the representation of introspections of preferences. Consider rules $r_\sharp$:

$$prefer(X, Y, kid) \leftarrow age\_interest(X, kids) \succcurlyeq_\sharp age\_interest(Y, kids),$$
$$package(X), package(Y)$$

and $r_\subseteq$:

$$request(X) \leftarrow age\_interest(X, teens) \succcurlyeq_\supseteq age\_interest(X, adults), package(X)$$

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 ASP$^{\mathrm{EP}}$. In section 4, we consider the relationship between ASP$^{\mathrm{EP}}$ and the strong introspection specification languages. In section 6, we explore the applications of ASP$^{\mathrm{EP}}$. We conclude in section 7 with some further discussion.

2 Answer Set Programming

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

$$l_1 \ or \ ... \ or \ l_k \leftarrow l_{k+1}, ..., l_m, not \ l_{m+1}, ..., not \ l_n$$

where the ls are literals of σ, *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 σ. 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 Π is a set of rules of the form

$$l_1 \ or \ ... \ or \ l_k \leftarrow e_1, ..., e_m, s_1, ..., s_n.$$

where $k \geq 0$, $m \geq 0$, $n \geq 0$, the ls are literals in first order logic language and are called objective literals here, es are extended literals which are 0-place connectives $\top$ and $\bot$, or objective literals possibly preceded by *not*, ss are subjective literals of the form $e \succcurlyeq_x e'$ or $e \not\succcurlyeq_x e'$ where e and e' are extended literals and $x \in \{\sharp, \supseteq\}$. 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 $sl(\Pi)$ to denote the set of subjective literals appearing in Π.

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 $e \succcurlyeq_x e'$ and $e' \not\succcurlyeq_x e$, and use $e \approx_x e'$ to denote the preferential indifference that can be expressed by the conjunction of

$e \not\succ_x e'$ and $e' \not\succ_x e$, and use $e \equiv_x e'$ to denote the preferential equivalence that can be expressed by the conjunction of $e \succcurlyeq_x e'$ and $e' \succcurlyeq_x e$.

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$^{\text{EP}}$ structure of W where $w \in W$. W is a model of a program Π if for each rule r in Π, r is satisfied by every pointed ASP$^{\text{EP}}$ structure of W. The notion of satisfiability denoted by $\models_{\text{ep}}$ is defined below.

- $(W, w) \models_{\text{ep}} \top$
- $(W, w) \not\models_{\text{ep}} \bot$
- $(W, w) \models_{\text{ep}} l$ if $l \in w$
- $(W, w) \models_{\text{ep}} \textit{not } l$ if $l \notin w$
- $(W, w) \models_{\text{ep}} e \succcurlyeq_\sharp e'$ if $|\{w \in W : (W, w) \models_{\text{ep}} e\}| \geq |\{v \in W : (W, v) \models_{\text{ep}} e'\}|$
- $(W, w) \models_{\text{ep}} e \succcurlyeq_\supseteq e'$ if $\{w \in W : (W, w) \models_{\text{ep}} e\} \supseteq \{v \in W : (W, v) \models_{\text{ep}} e'\}$
- $(W, w) \models_{\text{ep}} e \not\succcurlyeq_x e'$ if $(W, w) \not\models_{\text{ep}} e \succcurlyeq_x e'$, $x \in \{\sharp, \supseteq\}$

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

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

The satisfiability of a subjective literal does not depend on a specific belief set w in W, hence we can simply write $W \models_{\text{ep}} s$ if $(W, w) \models_{\text{ep}} s$ and say the subjective literal s is satisfied by W, and we can simply write $W \not\models_{\text{ep}} s$ if $(W, w) \not\models_{\text{ep}} s$ 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. $W \models_{\text{ep}} e \succ_x e' \Longrightarrow W \models_{\text{ep}} e' \not\succ_x e$
- $\succ_\sharp$ Inescapability. $W \models_{\text{ep}} e \succ_\sharp e'$,$W \models_{\text{ep}} e'' \not\succ_\sharp e' \Longrightarrow W \models_{\text{ep}} e \succ_\sharp e''$
- $\succ_x$ Transitivity. $W \models_{\text{ep}} e \succ_x e'$,$W \models_{\text{ep}} e' \succ_x e'' \Longrightarrow W \models_{\text{ep}} e \succ_x e''$
- $\succ_x$ Irreflexivity. $W \models_{\text{ep}} e \not\succ_x e$
- $\approx_x$ Reflexivity. $W \models_{\text{ep}} e \approx_x e$
- $\approx_x$ Symmetry. $W \models_{\text{ep}} e \approx_x e' \Longrightarrow W \models_{\text{ep}} e' \approx_x e$
- $\approx_\sharp$ Transitivity.$W \models_{\text{ep}} e \approx_\sharp e'$,$W \models_{\text{ep}} e' \approx_\sharp e'' \Longrightarrow W \models_{\text{ep}} e \approx_x e''$
- $\succ_\sharp$ R-Analogy. $W \models_{\text{ep}} e \succ_\sharp e'$,$W \models_{\text{ep}} e' \approx_\sharp e'' \Longrightarrow W \models_{\text{ep}} e \succ_\sharp e''$
- $\succ_\sharp$ L-Analogy. $W \models_{\text{ep}} e \approx_\sharp e'$,$W \models_{\text{ep}} e' \succ_\sharp e'' \Longrightarrow W \models_{\text{ep}} e' \succ_\sharp e''$

where $x \in \{\sharp, \supseteq\}$.

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

- $W \models_{\text{ep}} e \succcurlyeq_x e$
- $W \models_{\text{ep}} \top \succcurlyeq_x e$

- $W \models_{\text{ep}} e \succcurlyeq_x \bot$
- $W \models_{\text{ep}} e \not\succcurlyeq_{\supseteq} e^{not}$

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 Π be a disjunctive logic program, the candidate world view of Π is the non-empty set of all its answer sets, written as $AS(\Pi)$.

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

1. removing from Π all rules containing subjective literals not satisfied by W.
2. removing all other occurrences of subjective literals of the form $e \succcurlyeq_x e$ or $\top \succcurlyeq_x e$ or $e \succcurlyeq_x \bot$ or $e \not\succcurlyeq_{\supseteq} e^{not}$.
3. replacing all other occurrences of subjective literals of the form $e \succcurlyeq_x \top$ by e.
4. replacing all other occurrences of subjective literals of the form $\bot \succcurlyeq_x e$ by e^{not}.
5. replacing other occurrences of subjective literals of the form $e_1 \succcurlyeq_x e_2$ or $e_1 \not\succcurlyeq_x e_2$ 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, W is a candidate world view of Π if W is a candidate world view of Π^W.

We use $cwv(\Pi)$ to denote the set of candidate world views of an ASP$^{\text{EP}}$ program Π. Π^W is said to be the *reduct* of Π with respect to 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 $e \succcurlyeq_x e$ and $\top \succcurlyeq_x e$ and $e \succcurlyeq_x \bot$ and $e \not\succcurlyeq_{\supseteq} e^{not}$ 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 $e \succcurlyeq_x \top$ 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 $\bot \succcurlyeq_x e$ 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 $e_1 \succcurlyeq_x e_2$ may be true or not with regard to each belief set.

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

- $W \in cwv(\Pi)$
- Minimizing preferences principle: $\nexists V \in cwv(\Pi)(\{\bar{s}|s \in sl(\Pi) \wedge V \models_{ep} \bar{s}\} \supset \{\bar{s}|s \in sl(\Pi) \wedge W \models_{ep} \bar{s}\})$

where $\bar{s}$ is $e \succcurlyeq_x e'$ if s is $e \not\succcurlyeq_x e'$, and $\bar{s}$ is $e \not\succcurlyeq_x e'$ if s is $e \succcurlyeq_x e'$.

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

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

▶ **Example 6.** Consider $\Pi = \Pi_{\text{ep}} \cup \{r_\sharp, r_\supseteq\}$ where Π_{ep} and $r_\sharp$ and $r_\supseteq$ are given in section 1. It is easy to see that Π has an unique world view containing nine belief sets:

{*prefer(2,1),prefer(3,1),request(1),request(3),package(1),age_interest(1,kids), age_interest(1,adults),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(1),age_interest(1,kids), age_interest(1,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(1),age_interest(1,adults), age_interest(1,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(2),age_interest(2,kids), age_interest(2,adults),age_interest(2,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(2),age_interest(2,kids), age_interest(2,adults),age_interest(2,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(2),age_interest(2,adults), age_interest(2,kids),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(3),age_interest(3,kids), age_interest(3,adults),age_interest(3,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),package(3),age_interest(3,kids), age_interest(3,adults),age_interest(3,teens),...*}
{*prefer(2,1),prefer(3,1),request(1),request(3),age_interest(3,teens), age_interest(1,kids),...*}

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

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

that states a simple ordered-based choice strategy, then we can get $\Pi \vdash_{\text{ep}} buy(3)$ 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^{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^{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

Table 2 Modal Reduct in ASP$^{\mathrm{KM}}$.

subjective literal s	if $W \models_{\mathrm{km}} s$	$W \not\models_{\mathrm{km}} s$
Kl	replace Kl with l	delete the rule
not Kl	remove not Kl	replace not Kl with not l
Ml	remove Ml	replace Ml with not not l
not Ml	replace not Ml with not l	delete the rule

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$^{\mathrm{NOT}}$), and defined its world views by minimizing the knowledge. ASP$^{\mathrm{NOT}}$ 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$^{\mathrm{KM}}$ logic programs in [16] where the most recent version of ASP$^{\mathrm{KM}}$ is defined, and a special kind of ASP$^{\mathrm{NOT}}$ programs can be viewed as ASP$^{\mathrm{EP}}$ programs.

4.1 Relation to ASP$^{\mathrm{KM}}$

An ASP$^{\mathrm{KM}}$ program is a set of rules of the form h_1 *or* ... *or* $h_k \leftarrow b_1, ..., b_m$ 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$^{\mathrm{KM}}$ program **KM-world view**. Let W be a non-empty collection of consistent sets of ground objective literals, W is a **KM-world view** of an ASP$^{\mathrm{KM}}$ program Π if $W = AS(\Pi_W)$ where Π_W is a disjunctive logic program obtained using *Modal Reduct* as showed in Table 2.

In ASP$^{\mathrm{KM}}$, the notion of satisfiability is defined from $\models_{\mathrm{km}}$ relationship below.

- $< W, w >\models_{\mathrm{km}} l$ if $l \in w$
- $< W, w >\models_{\mathrm{km}}$ *not* l if $l \notin w$
- $< W, w >\models_{\mathrm{km}}$ Kl if $\forall v \in W : l \in v$
- $< W, w >\models_{\mathrm{km}}$ *not* Kl if $\exists v \in W : l \notin v$
- $< W, w >\models_{\mathrm{km}}$ Ml if $\exists v \in W : l \in v$
- $< W, w >\models_{\mathrm{km}}$ *not* Ml if $\forall v \in W : l \notin v$

▶ **Definition 7.** Given an ASP$^{\mathrm{KM}}$ program Ω, an ASP$^{\mathrm{EP}}$ program is called a *KM-EP-Image* of Ω, denoted by $KM - EP - I(\Omega)$, if it is obtained by

- Replacing all occurrences of literals of the form $K\ l$ in Π by $l \succeq_{\sharp} \top$.
- Replacing all occurrences of literals of the form $M\ l$ in Π by *not* $l \not\succeq_{\sharp} \top$ and *not not* l[4] respectively.
- Replacing all occurrences of literals of the form *not* $K\ l$ in Π by $l \not\succeq_{\sharp} \top$ and *not* l respectively.
- Replacing all occurrences of literals of the form *not* $M\ l$ in Π by *not* $l \succeq_{\sharp} \top$.

▶ **Theorem 8.** *Let Ω be an ASP^{KM} program, and Π be the ES-EP-Image of Ω, and W be a non-empty collection of consistent sets of ground objective literals, W is a candidate world view of Π iff W is a KM-world view of Ω.*

[4] Here, we view *not not* l as a representation of *not* l' where we have $l' \leftarrow$ *not* l and l' is a fresh literal. It is worthwhile to note that CLINGO is able to deal with *not not*.

▶ **Example 9.** Consider an ASP$^{\text{KM}}$ program Ω: $p \leftarrow \text{M}\ p$. Ω has an unique KM-world view $\{\{p\}\}$. Its ES-EP-Image Π contains two rules

$$p \leftarrow not\ p \not\succcurlyeq_{\sharp} \top \qquad p \leftarrow\ not\ not\ p$$

Then, the reduct $\Pi^{\{\{p\}\}}$ contains five rules

$$p \leftarrow p, \top \qquad p \leftarrow not\ p, \top \qquad p \leftarrow p, \bot \qquad p \leftarrow not\ p, \bot \qquad p \leftarrow\ not\ not\ p$$

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

4.2 Relation to ASP$^{\text{NOT}}$

Here, we consider the ASP$^{\text{NOT}}$ program that is a set of the rules of the form $l_1\ or\ ...\ or\ l_k \leftarrow e_1, ..., e_m, s_1, ..., s_n$ where $k \geq 0$, $m \geq 0$, $n \geq 0$, l_i is an objective literal, e_i is an extended literal, s_i is a subjective literal of the form NOT e or *not* NOT e. For distinguishment, we call the world view of an ASP$^{\text{NOT}}$ 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$^{\text{NOT}}$ program Π if $W = AS(\Pi_W)$ 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 *not* F. In ASP$^{\text{NOT}}$, the notion of satisfiability of a subjective formula NOT F is defined from $\models_{\text{NOT}}$ relationship

$$< W, w > \models_{\text{NOT}} \text{NOT } F \text{ if } \exists v \in W : v \not\models_{\text{GLP}} F$$

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$^{\text{NOT}}$ 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$^{\text{NOT}}$ program Ω, an ASP$^{\text{EP}}$ 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 *not NOT* e in Ω by $e \succcurlyeq_{\sharp} \top$.
- Replacing all occurrences of literals of the form *NOT* e in Ω by $e \not\succcurlyeq_{\sharp} \top$ and *not* e respectively.

▶ **Theorem 11.** *Let Ω be an ASP^{NOT} 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$^{\text{NOT}}$ program from [22] that contains two rules

$$innocent(john) | guilty(john) \qquad innocent(john) \leftarrow \text{NOT } guility(john)$$

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

$$innocent(john) | guilty(john)$$
$$innocent(john) \leftarrow guilty(john) \not\succcurlyeq_{\sharp} \top$$
$$innocent(john) \leftarrow not\ guility(john)$$

and a unique world view $\{\{innocent(john)\}\}$.

5 Applications

Consider the relationship between ASP$^{\mathrm{EP}}$ and the languages of strong introspections mentioned in section 5, ASP$^{\mathrm{EP}}$ is potential in dealing with some important issues. In this section, we illustrate the use of ASP$^{\mathrm{EP}}$ 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:

$$pigeon(X) \text{ } or \text{ } raven(X) \text{ } or \text{ } swallow(X) \text{ } sparrow(X) \leftarrow commonBird(X)$$
$$behavior(X, migratory) \leftarrow swallow(X)$$
$$behavior(X, resident) \leftarrow pigeon(X)$$
$$behavior(X, resident) \leftarrow raven(X)$$
$$behavior(X, resident) \leftarrow sparrow(X)$$

Then, given a fact f_t:

$$commonBird(tom)$$

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

$$behavior(X, resident) \leftarrow behavior(X, resident) \succ_\sharp behavior(X, migratory), bird(X)$$
$$behavior(X, migratory) \leftarrow behavior(X, migratory) \succ_\sharp behavior(X, resident), bird(X)$$
$$behavior(X, unknown) \leftarrow behavior(X, migratory) \approx_\sharp behavior(X, resident), bird(X)$$

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 $PM \cup \{f_t, r_r, r_m, r_u\}$ gives answer $behavior(tom, resident)$ to the query, that is

$$PM \cup \{f_t, r_r, r_m, r_u\} \vdash_{\mathrm{ep}} behavior(tom, resident)$$

5.2 Modeling the Monty Hall Problem

We will use ASP$^{\mathrm{EP}}$ 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: *Constestant switches if SWITCH can bring more wins than STAY, Constestant stays if STAY can bring more wins than SWITCH.*

Table 3 Possible Results of MHP.

Keys are in box	Contestant choose box	Monty can open box	Contestant switches	Results
1	1	2 or 3	2 or 3	loses
1	2	3	1	wins
1	3	2	1	wins
2	1	3	2	wins
2	2	1 or 3	1 or 3	loses
2	3	1	2	wins
3	1	2	3	wins
3	2	1	3	wins
3	3	1 or 2	1 or 2	loses

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

$box(1)$
$box(2)$
$box(3)$
$1\{choose\_box(X) : box(X)\}1$
$1\{key\_in\_box(X) : box(X)\}1$
$can\_open\_box(X) \leftarrow box(X), not\ choose\_box(X), not\ key\_in\_box(X)$
$win\_by\_switch \leftarrow choose\_box(X), not\ key\_in\_box(X)$
$win\_by\_stay \leftarrow choose\_box(X), key\_in\_box(X)$

Represent the idea in the solution by two rules r_1 :

$$switch \leftarrow win\_by\_switch \succcurlyeq_\sharp win\_by\_stay, win\_by\_stay \not\succcurlyeq_\sharp win\_by\_switch$$

and r_2:

$$stay \leftarrow win\_by\_stay \succcurlyeq_\sharp win\_by\_switch, win\_by\_switch \not\succcurlyeq_\sharp win\_by\_stay$$

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

▶ **Theorem 13.** *$MHP \cup \{r_1, r_2\} \vdash_{ep} switch$ and $MHP \cup \{r_1, r_2\} \nvdash_{ep} stay$.*

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$^{\text{EP}}$ and other well developed formalisms of preferences.

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

References

1 Marcello Balduccini and Michael Gelfond. Logic Programs with Consistency-Restoring Rules. In *International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series*, pages 9–18, 2003.

2 Ronen I. Brafman and Carmel Domshlak. Preference Handling - An Introductory Tutorial. *AI Magazine*, 30(1):58–86, 2009. URL: `http://www.aaai.org/ojs/index.php/aimagazine/article/view/2114`.

3 Gerhard Brewka. Answer Sets and Qualitative Optimization. *Logic Journal of the IGPL*, 14(3):413–433, 2006. `doi:10.1093/jigpal/jzl017`.

4 Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Logic Programs with Ordered Disjunction. *Computational Intelligence*, 20(2):335–357, 2004. `doi:10.1111/j.0824-7935.2004.00241.x`.

5 Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Answer Set Optimization. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 867–872, 2003. URL: `http://ijcai.org/Proceedings/03/Papers/125.pdf`.

6 Luis Fariñas Del Cerro, Andreas Herzig, and Ezgi Iraz Su. Epistemic equilibrium logic. In *Proc. 24th Int. Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 2964–2970, 2015.

7 James P. Delgrande, Torsten Schaub, Hans Tompits, and Kewen Wang. A Classification and Survey of Preference Handling Approaches in Nonmonotonic Reasoning. *Computational Intelligence*, 20(2):308–334, 2004. `doi:10.1111/j.0824-7935.2004.00240.x`.

8 Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in AI: An overview. *Artif. Intell.*, 175(7-8):1037–1052, 2011. `doi:10.1016/j.artint.2011.03.004`.

9 Wolfgang Faber, Gerald Pfeifer, Nicola Leone, Tina Dell'armi, and Giuseppe Ielpa. Design and Implementation of Aggregate Functions in the DLV System. *Theory Pract. Log. Program.*, 8(5-6):545–580, November 2008. `doi:10.1017/S1471068408003323`.

10 Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven Answer Set Solving: From Theory to Practice. *Artif. Intell.*, 187–188:52–89, August 2012. `doi:10.1016/j.artint.2012.04.001`.

11 Michael Gelfond. Strong Introspection. In Thomas L. Dean and Kathleen McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA, USA, July 14-19, 1991, Volume 1.*, pages 386–391. AAAI Press / The MIT Press, 1991. URL: `http://www.aaai.org/Library/AAAI/1991/aaai91-060.php`.

12 Michael Gelfond. Logic Programming and Reasoning with Incomplete Information. *Ann. Math. Artif. Intell.*, 12(1-2):89–116, 1994. `doi:10.1007/BF01530762`.

13 Michael Gelfond. New semantics for epistemic specifications. In *Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer, 2011.

14 Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*. Cambridge Unversity Press, 2014.

15 Judy Goldsmith and Ulrich Junker. Preference Handling for Artificial Intelligence. *AI Magazine*, 29(4):9–12, 2008. URL: `http://www.aaai.org/ojs/index.php/aimagazine/article/view/2180`.

16 Patrick Kahl, Richard Watson, Michael Gelfond, and Yuanlin Zhang. A Refinement of the Language of Epistemic Specifications. *Journal of Logic and Computation*, 2015. URL: `10.1093/logcom/exv065`.

17 Pascal Nicolas, Laurent Garcia, Igor Stéphan, and Claire Lefèvre. Possibilistic uncertainty handling for answer set programming. *Ann. Math. Artif. Intell.*, 47(1-2):139–181, 2006. `doi:10.1007/s10472-006-9029-y`.

18 Gabriella Pigozzi, Alexis Tsoukiàs, and Paolo Viappiani. Preferences in artificial intelligence. *Ann. Math. Artif. Intell.*, 77(3-4):361–401, 2016. `doi:10.1007/s10472-015-9475-5`.

19 Chiaki Sakama and Katsumi Inoue. Prioritized logic programming and its application to commonsense reasoning. *Artif. Intell.*, 123(1-2):185–222, 2000. `doi:10.1016/S0004-3702(00)00054-0`.

20 Torsten Schaub and Kewen Wang. A semantic framework for preference handling in answer set programming. *TPLP*, 3(4-5):569–607, 2003. `doi:10.1017/S1471068403001844`.

21 Steve Selvin. A problem in probability (letter to the editor). *American Statistician*, 29:67, 1975.

22 Yi-Dong Shen and Thomas Eiter. Evaluating epistemic negation in answer set programming. *Artificial Intelligence*, 237:115–135, 2016.

23 Yidong Shen, Kewen Wang, and Thomas Eiter. FLP answer set semantics without circular justifications for general logic programs. *Artificial Intelligence*, 213:1–41, 2014.

24 Mirosław Truszczyński. Revisiting epistemic specifications. In *Logic programming, knowledge representation, and nonmonotonic reasoning*, pages 315–333. Springer, 2011.

25 G. H. Von Wright. The logic of preference reconsidered. *Theory and Decision*, 3(2):140–169, 1972. `doi:10.1007/BF00141053`.