

A Refinement of the Language of Epistemic Specifications

Patrick Kahl, Richard Watson, Michael Gelfond, and Yuanlin Zhang

Department of Computer Science, Texas Tech University
{patrick.kahl, richard.watson, michael.gelfond, y.zhang}@ttu.edu

Abstract. In this paper we present a new version of the language of Epistemic Specifications. The goal is to simplify and improve the intuitive and formal semantics of the language. We describe an algorithm for computing solutions of programs written in this new version of the language. The new semantics is illustrated by a number of examples, including an Epistemic Specifications-based framework for conformant planning.

Keywords: Epistemic Specifications, Epistemic Logic Programs, Logic Programming, Introspective Reasoning, Answer Set Programming extensions, Conformant Planning

1 Introduction

We define a new version of the language of *Epistemic Specifications*—an extension of the language of answer set programming (ASP)—that was introduced by Gelfond in the early 1990s[12,13]. The language of Epistemic Specifications allows for introspective reasoning through the use of modal operators K (“known”) and M (“may be true”). Even though the original version of the language allowed formalizing certain types of knowledge not expressible in ASP, its semantics was not fully satisfactory. The difficulty was related to issues with the formalization of the *rationality principle* which states that *a rational agent should believe only what he is forced to believe*. The semantics of the language didn’t seem to adequately capture this principle for programs with recursion through modal operators. The problem was partially addressed in [14] where some progress was made with respect to the semantics of programs with recursion through the K operator. The approach was, however, not sufficient for programs with recursion through M. The version of the language presented in this paper addresses this issue.

In the next section, we provide some background information about the language. In section 3, we give the syntax and semantics for the new version. We then show comparisons between the new and old versions in section 4. An algorithm for an Epistemic Specifications solver is given in section 5. We next look at applications in section 6, including a framework for a conformant planner that takes advantage of the added features of the language. Finally, we describe related work, conclusions, and future work.

2 Background

The first paper[12] on Epistemic Specifications illustrated the need for a new formalism by considering the following problem.

Example 1. [Scholarship Eligibility Problem]

Consider the following three ASP rules (with rule labels added for later reference)

$$\begin{aligned} R_1: & \text{eligible}(X) \leftarrow \text{highGPA}(X). \\ R_2: & \text{eligible}(X) \leftarrow \text{minority}(X), \text{fairGPA}(X). \\ R_3: & \neg\text{eligible}(X) \leftarrow \neg\text{fairGPA}(X), \neg\text{highGPA}(X). \end{aligned}$$

used by a certain college to determine eligibility of students for scholarships, where the variable X ranges over a given set of students. A fourth rule—given here in plain text—is that *a student should be interviewed by the scholarship committee if his/her eligibility cannot be determined using the prior three rules.*

A reasonable attempt to express this fourth rule in ASP is as follows:

$$R_4: \text{interview}(X) \leftarrow \text{not eligible}(X), \text{not } \neg\text{eligible}(X).$$

However, when used with a database containing incomplete information, the resulting program is unable to correctly determine if a student is supposed to be interviewed. This is illustrated by considering the following disjunctive rule

$$R_5: \text{fairGPA}(\text{mike}) \text{ or } \text{highGPA}(\text{mike}).$$

representing information from a database that Mike's GPA is either fair or high. Note that his minority status is not given and is therefore unknown. Together, rules R_1 – R_5 comprise a program having the following two answer sets:

$$\begin{aligned} A_1 &= \{\text{highGPA}(\text{mike}), \text{eligible}(\text{mike})\}, \text{ and} \\ A_2 &= \{\text{fairGPA}(\text{mike}), \text{interview}(\text{mike})\}. \end{aligned}$$

Although the program correctly represents Mike's eligibility for a scholarship as indeterminate, it fails to entail that Mike should be interviewed. A general representation was unable to be found for solving this problem using ASP.

Gelfond's remedy to this problem was to extend ASP by introducing modal operators, providing for more powerful introspective reasoning with respect to *all* answer sets (the K modal operator) or *some* answer set (the M modal operator). Semantically, programs in the language of Epistemic Specifications define possible collections of belief sets of a rational agent—called *world views*—associated with the program.

The world view of a program without modal operators coincides with the collection of its answer sets if taken as an ASP program. If a program has modal operators, a world view W satisfies *subjective literal* K ℓ if literal ℓ belongs to all the belief sets in W , and *subjective literal* M ℓ if ℓ belongs to at least one belief set in W . For example, $\{\{p, r\}, \{q, r\}\}$ is the world view of the following program:

$$\begin{aligned} p &\text{ or } q. \\ r &\leftarrow \text{M } p. \end{aligned}$$

The language of Epistemic Specifications provides a natural way to deal with the problem described in Example 1.

Example 2. [Scholarship Eligibility Problem Revisited]

The fourth rule can be expressed in the language of Epistemic Specifications as

$$R_6: \text{interview}(X) \leftarrow \text{not K eligible}(X), \text{not K } \neg\text{eligible}(X).$$

The epistemic logic program consisting of rule R_6 together with rules R_1 – R_3 and R_5 from Example 1 has the world view $W = \{A_1, A_2\}$ where

$$A_1 = \{highGPA(mike), eligible(mike), interview(mike)\}, \text{ and} \\ A_2 = \{fairGPA(mike), interview(mike)\}.$$

Notice that $interview(mike)$ is in every belief set of the world view of this program; thus, we now have a program that correctly entails that Mike should be interviewed.

Now let's consider the following program:

$$p \leftarrow K p.$$

Under the original semantics, this program has two world views: $\{\{\}\}$ and $\{\{p\}\}$. We believe a rational agent should not accept $\{\{p\}\}$ as a world view of this program as support for p is clearly circular and belief in $K p$ should require stronger conviction than that given (e.g., since the one-rule program $p \leftarrow p$ does not force belief in p , it is clear that $p \leftarrow K p$ should not since $K p$ requires more conviction to accept than p).

Gelfond updated the semantics in [14], fixing the problem with recursion through K such that $\{\{\}\}$ was the only world view of $p \leftarrow K p$; however, the following program

$$p \leftarrow M p.$$

still had two world views: $\{\{\}\}$ and $\{\{p\}\}$. We couldn't find a justification for this result compatible with the rationality principle.

3 Syntax and Semantics of Epistemic Specifications—New Version

In what follows we will distinguish between different versions of Epistemic Specifications by the years of their introduction. The new version will be referred to as *ES2014*.

3.1 Syntax of ES2014

The language of Epistemic Specifications consists of the usual first-order signature, ASP logical connectives (`not`, `or`, `¬`, etc.), and modal operators K and M . An *objective literal* is the same as a literal in ASP. A *subjective literal* is an expression of the form $K \ell$ or $M \ell$ where ℓ is an objective literal. A *default-negated literal* is an expression of the form `not` g where g is an objective or subjective literal. An *extended literal* is an objective, subjective, or default-negated literal. A rule is of the form

$$\ell_1 \text{ or } \dots \text{ or } \ell_k \leftarrow e_1, \dots, e_n.$$

where $k \geq 0$, $n \geq 0$, ℓ s are objective literals, and e s are extended literals. A program in this language, called an *epistemic logic program*, is a set of rules.

3.2 Semantics of ES2014

Note: In the following definitions, ℓ is used to denote a ground objective literal.

By an *ES structure*, W , we mean a non-empty collection of consistent sets of ground objective literals. A pair $\langle A, W \rangle$ is called a *pointed ES structure* of W where $A \in W$. To define the semantics of Epistemic Specifications, we first define the notion of a model of an epistemic logic program Π . W is a *model* of Π if for each rule R in Π , R is satisfied by every pointed ES structure of W . The notion of satisfiability is defined below.

Definition 1. [*When an Extended Literal Is Satisfied*]

Let W be an ES structure and $\langle A, W \rangle$ be a pointed ES structure of W .

- $\langle A, W \rangle \models \ell$ if $\ell \in A$.
- $\langle A, W \rangle \models \text{not } \ell$ if $\ell \notin A$.
- $\langle A, W \rangle \models K \ell$ if $\forall S \in W : \ell \in S$.
- $\langle A, W \rangle \models \text{not } K \ell$ if $\exists S \in W : \ell \notin S$.
- $\langle A, W \rangle \models M \ell$ if $\exists S \in W : \ell \in S$.
- $\langle A, W \rangle \models \text{not } M \ell$ if $\forall S \in W : \ell \notin S$.

Note that the satisfiability of an objective literal does not depend on W and the satisfiability of a subjective literal does not depend on A . For example, if $\langle A, W \rangle \models \ell$ we can simply say that $A \models \ell$; likewise, if $\langle A, W \rangle \models K \ell$ we can simply say that $W \models K \ell$.

Definition 2. [*When a Rule Is Satisfied*]

Let Π be a ground epistemic logic program, W be an ES structure, $\langle A, W \rangle$ be a pointed ES structure of W , and R be a rule in Π . $\langle A, W \rangle \models R$ if

- $\exists \ell \in \text{head}(R) : \langle A, W \rangle \models \ell$, or
- $\exists e \in \text{body}(R) : \langle A, W \rangle \not\models e$

where $\text{head}(R)$ denotes the set of objective literals in the head of R and $\text{body}(R)$ denotes the set of extended literals in the body of R .

The following definition embodies the core of the new semantics for ES2014.

Definition 3. [*Modal Reduct*]

Let Π be a ground epistemic logic program and W be an ES structure. The *modal reduct* of Π with respect to W , denoted by Π^W , is the ASP program (with *nested expressions*¹) obtained from Π by replacing/removing subjective literals and deleting rules according to the following table.

s	if $W \models s$	if $W \not\models s$
$K \ell$	replace $K \ell$ with ℓ	delete the rule
$\text{not } K \ell$	remove $\text{not } K \ell$	replace $\text{not } K \ell$ with $\text{not } \ell$
$M \ell$	remove $M \ell$	replace $M \ell$ with $\text{not not } \ell$
$\text{not } M \ell$	replace $\text{not } M \ell$ with $\text{not } \ell$	delete the rule

Definition 4. [*World View*]

W is a *world view* of Π if $W = \text{AS}(\Pi^W)$ where $\text{AS}(\Pi^W)$ denotes the set of all answer sets of Π^W . Π is said to be *inconsistent* if Π has no world view.

¹ By *nested expressions* we mean those of the form $\text{not not } \ell$ in the body of a rule as defined in [17]. For ease of presentation, we may use *ASP* to mean *ASP with nested expressions*.

Not only does this new definition retain the “fix” for unintended world views due to recursion through K (as provided in ES2011), it also provides a reasonable solution to the problem of unintended world views due to recursion through M. The next example demonstrates this important new feature of our new definition.

Example 3.

Let Π be the following epistemic logic program.

$$p \leftarrow M p.$$

Consider the ES structure $W_1 = \{\{\}\}$. Π^{W_1} is

$$p \leftarrow \text{not not } p.$$

which has two answer sets, $\{\}$ and $\{p\}$; thus, since $W_1 \neq \text{AS}(\Pi^{W_1})$, W_1 is *not* a world view of Π . Now consider the ES structure $W_2 = \{\{p\}\}$. Π^{W_2} is

$$p.$$

whose answer set is $\{p\}$; thus, since $W_2 = \text{AS}(\Pi^{W_2})$, W_2 is a world view of Π .

It is worth noting that, per our original intuition, we believed W_1 should have been the only world view of Π . New insights into the nature of modal operators forced us to change our view. In the next section, we give a short description of our new intuition.

3.3 Preference Order for Extended Literals and New Intuition

It was observed by looking at the definitions for satisfiability that a rational agent should find it easier to accept certain extended literals over others. This is clear when we look at the fact that, for example, given a pointed ES structure $\langle A, W \rangle$, in order to establish $K \ell$, it must be demonstrated that ℓ belongs to *all* belief sets in W . To establish ℓ , it must be demonstrated that ℓ belongs to a *particular* belief set in W , namely A . But to establish $M \ell$, it is sufficient to demonstrate that ℓ belongs to *some* belief set in W . This implies a preference order as illustrated below.

hardest to accept/ highest conviction ↑ ↓ easiest to accept/ lowest conviction $K \ell$ ℓ $M \ell$ $\text{not } M \ell$ $\text{not } \ell$ $\text{not } K \ell$		Example Programs Illustrating the Preference Order for Default-negated Literals		
		EXAMPLE A: $p \leftarrow \text{not } M q.$ $q \leftarrow \text{not } p.$ World view: $\{\{q\}\}$	EXAMPLE B: $p \leftarrow \text{not } M q.$ $q \leftarrow \text{not } K p.$ World view: $\{\{q\}\}$	EXAMPLE C: $p \leftarrow \text{not } q.$ $q \leftarrow \text{not } K p.$ World view: $\{\{q\}\}$

We believe this to be an important observation, one that is key to the development of an intuitive understanding of the semantics of our new definition, and for our acceptance of $\{\{p\}\}$ as the one and only world view of $p \leftarrow M p$. In order to formalize our intuition about the preference of ℓ with respect to $M \ell$, the definition of the reduct uses a translation of $M \ell$ into $\text{not not } \ell$ rather than ℓ when $M \ell$ is not satisfied.

The next example demonstrates another feature of the new semantics in that replacement of $M \ell$ by $\text{not not } \ell$ when $M \ell$ is not satisfied allows acceptance of a world view where none existed under any of the semantics of previous language versions.

Example 4.

Let Π be the following epistemic logic program.

$$\begin{aligned} p &\text{ or } q. \\ p &\leftarrow \text{M } q. \end{aligned}$$

Consider the ES structure $W = \{\{p\}\}$. Π^W is

$$\begin{aligned} p &\text{ or } q. \\ p &\leftarrow \text{not not } q. \end{aligned}$$

whose answer set is $\{p\}$; thus, since $W = \text{AS}(\Pi^W)$, $\{\{p\}\}$ is a world view of Π .

In the next section we provide more examples to demonstrate the ensuing effects of the new semantics.

4 Comparison with Previous Language Version

The modal reduct in Gelfond's 1994 definition[13] (referred to henceforth as *ES1994*) simply removed all satisfied subjective literals from the bodies of rules and removed any rules containing unsatisfied subjective literals. ES1994 also allowed empty world views and world views with an inconsistent belief set. The syntax of ES2014 differs from ES1994 by changing $\neg K \ell$ and $\neg M \ell$ to $\text{not } K \ell$ and $\text{not } M \ell$, respectively. The table below shows world views and respective modal reducts of various example programs for ES1994 and ES2014.

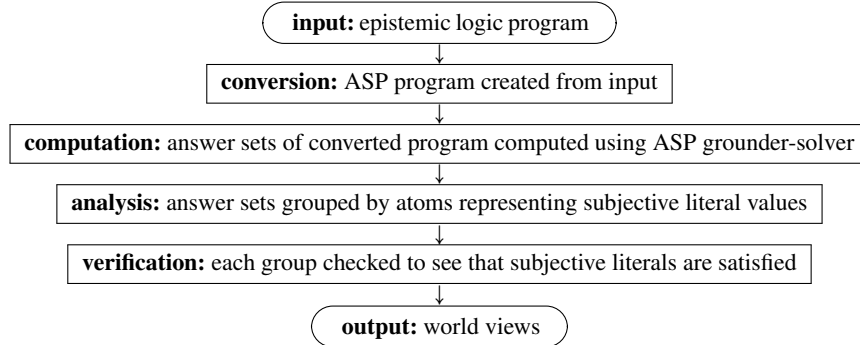
Π	W	Π^W [ES1994]	Π^W [ES2014]	Comment
$p \leftarrow \text{not } p, K p.$	$\{\}$	$p \leftarrow \text{not } p.$	[n/a]	empty world view
	World view?	YES	NO	
	$\{\{\}\}$			
	World view?	YES	YES	
$p.$ $\neg p.$	$\{\{p, \neg p\}\}$	$p.$ $\neg p.$	[n/a]	inconsistent belief set
	World view?	YES	NO	
$p \leftarrow K p.$ $\neg p \leftarrow K \neg p.$	$\{\{\}\}$			self-supported
	World view?	YES	YES	
	$\{\{p\}\}$	$p.$	$p \leftarrow p.$	
	World view?	YES	NO	self-supported
	$\{\{\neg p\}\}$	$\neg p.$	$\neg p \leftarrow \neg p.$	
	World view?	YES	NO	
	$\{\{p, \neg p\}\}$	$p.$ $\neg p.$	[n/a]	inconsistent belief set
	World view?	YES	NO	
$p \text{ or } q.$ $\neg p \leftarrow \text{not } M p.$	$\{\{p\}, \{q\}\}$	$p \text{ or } q.$	$p \text{ or } q.$	indirectly self-supported
	World view?	YES	YES	
—or, using ES1994 syntax—	$\{\{\neg p, q\}\}$	$p \text{ or } q.$	$p \text{ or } q.$	
$p \text{ or } q.$ $\neg p \leftarrow \neg M p.$		$\neg p.$	$\neg p \leftarrow \text{not } p.$	indirectly self-supported
	World view?	YES	NO	
$p \text{ or } q.$ $q \text{ or } r.$ $p \leftarrow M p.$	$\{\{p, q\}, \{p, r\}\}$	$p \text{ or } q.$ $q \text{ or } r.$ $p.$	$p \text{ or } q.$ $q \text{ or } r.$ $p.$	externally supported
	World view?	YES	YES	

Among the differences given between the new and previous semantics is that a world view is now required to be *non-empty* and composed of *consistent* sets of literals. It may be viewed that since the definition of a world view was dependent on ASP semantics that the consistency requirement for belief sets in a world view came indirectly as a result of the evolution of ASP semantics. (The non-empty world view requirement was first proposed by Truszczyński in [21].)

5 An Algorithm for Finding World Views of an ES2014 Program

We propose an algorithm that takes an epistemic logic program as its input and outputs the world views (if any) of the program. We first generate candidate solutions by creating an ASP program from the input that emulates all possible modal reducts with respect to combinations of truth assignments to distinct ground subjective literals of the input program. The answer sets of the ASP program are then computed and grouped into candidate world views based on associated subjective literal truth values. Finally, the candidates are checked by verifying that the associated subjective literals are all satisfied by the group of answer sets. Validated candidates are world views.

Outline of Algorithm for Finding World Views of an Epistemic Logic Program



For ease of presentation, k_l , $k0_l$, $k1_l$, m_l , $m0_l$, $m1_l$, or d_l will be used to denote the fresh atom obtained from literal l by prefixing it with k -, $k0$ -, $k1$ -, m -, $m0$ -, $m1$ -, or d - (respectively),² and with substitution of 0 for $\neg$ within the fresh atom implicitly understood if l is a classically-negated atom. For example, if $l = p(X)$ then $k0_l = k0_p(X)$, but if $l = \neg p(X)$ then $k0_l = k0_0p(X)$. An atom of the form k_l , $k0_l$, or $k1_l$ will be referred to as a *k-atom*, whereas an atom of the form m_l , $m0_l$, or $m1_l$ will be referred to as an *m-atom*. By a *k-/m-literal* we mean a k-/m-atom or its negation. If W is a collection of sets of literals, we denote by $W_{\setminus \text{KM}}$ the collection of sets from W with all k-/m-literals removed. An atom of the form d_l is used strictly as a device to achieve proper grounding of rules associated with a subjective literal of the form Kl , ensuring rule safety when l contains a variable. After grounding, atoms of the form d_l are removed. Likewise, $rbdy^+(R)$ —denoting the comma-delimited list of relevant positive literals from the body of rule R when R contains a subjective literal containing a variable—is used to achieve proper grounding and safety of added rules.

² We assume prefixes used here do not occur in Π . Other prefixes may be used to avoid conflicts.

In the following algorithm, $k1_l$ is to be intuitively understood as “ $K\ l$ is true” and $k0_l$ as “ $K\ l$ is false.” Respectively, $m1_l$ ($m0_l$) corresponds to “ $M\ l$ is true (false).”

Algorithm 1. [ELPSolver: Finds the World Views of an Epistemic Logic Program]

INPUT: an epistemic logic program Π with only safe rules and an equivalent finite ground representation
OUTPUT: the world views (if any) of Π

1. Create ASP program Π' from the rules of Π . Rules without subjective literals are left unchanged. For each rule R containing a subjective literal, replace the subjective literal and add new rules per the following table.

SUBJECTIVE LITERAL	REPLACE WITH	ADD RULES	NOTE: Items in brackets are added only if l contains a variable.
$K\ l$	$\text{not } \neg k_l, \ l$	$[d_l \leftarrow l.]$ $k1_l \leftarrow \text{not } k0_l[, \text{rbody}^+(R), d_l].$ $k0_l \leftarrow \text{not } k1_l[, \text{rbody}^+(R), d_l].$ $\neg k_l \leftarrow k0_l.$ $\neg k_l \leftarrow k1_l, \text{not } l.$	
$\text{not } K\ l$	$\neg k_l$	$k1_l \leftarrow \text{not } k0_l[, \text{rbody}^+(R)].$ $k0_l \leftarrow \text{not } k1_l[, \text{rbody}^+(R)].$ $\neg k_l \leftarrow k0_l.$ $\neg k_l \leftarrow k1_l, \text{not } l.$	
$M\ l$	m_l	$m1_l \leftarrow \text{not } m0_l[, \text{rbody}^+(R)].$ $m0_l \leftarrow \text{not } m1_l[, \text{rbody}^+(R)].$ $m_l \leftarrow m1_l.$ $m_l \leftarrow m0_l, \text{not not } l.$	
$\text{not } M\ l$	$\text{not } m_l$	$m1_l \leftarrow \text{not } m0_l[, \text{rbody}^+(R)].$ $m0_l \leftarrow \text{not } m1_l[, \text{rbody}^+(R)].$ $m_l \leftarrow m1_l.$ $m_l \leftarrow m0_l, \ l.$	

2. Compute Π'_{ground} (the ground version of Π') with an appropriate ASP grounder, then create Π''_{ground} from Π'_{ground} by removing each rule with an atom of the form d_l in its head along with every occurrence of d_l from the rest of the program, and then compute the answer sets of Π''_{ground} with an appropriate ASP solver.

NOTE: If there are no answer sets (i.e., Π''_{ground} is inconsistent), then Π has no world view. If Π has no subjective literals and Π''_{ground} is consistent, then the collection of answer sets of Π''_{ground} is the only world view of Π . In either case, the algorithm is finished and no further steps are performed.

3. Group the answer sets of Π''_{ground} by common k-/m-atoms of the form $k0_l$, $k1_l$, $m0_l$, and $m1_l$. Each group W of answer sets is said to *represent* (i.e., if its k-/m-literals are removed) a *candidate world view* of Π .
4. For each W representing a candidate world view of Π , check that the following conditions are met for all its k-/m-atoms:
 - (a) if $k1_l$ is in the sets of W , then l is in every set of W ;
 - (b) if $k0_l$ is in the sets of W , then l is missing from at least one set of W ;
 - (c) if $m1_l$ is in the sets of W , then l is in at least one set of W ; and
 - (d) if $m0_l$ is in the sets of W , then l is missing from every set of W .

W_{KM} is a *world view* of Π if conditions (a)–(d) above are met.

Theorem 1. *[The Algorithm Is Sound]³*

Given as input an epistemic logic program Π with only safe rules and an equivalent finite ground representation, if W is output by Algorithm 1 then W is a world view of Π .

Theorem 2. *[The Algorithm Is Complete]*

Given as input an epistemic logic program Π with only safe rules and an equivalent finite ground representation, if W is a world view of Π then W is output by Algorithm 1.

6 Applications (Conformant Planning)

There are many potential applications of the language of Epistemic Specifications in domains with incomplete information. These include:

- conformant planning;
- autonomous robot control;
- policy management;
- assisting legal professionals (e.g., logical analysis of criminal evidence); and
- any application for which ASP is currently used but where the additional power of Epistemic Specifications might be needed due to incomplete information.

In this section we briefly discuss one of these possibilities, that of conformant planning. *Conformant planning*[19] is the problem of finding a plan (a sequence of actions) that is guaranteed to achieve a goal in spite of incomplete knowledge about the initial state. Research on the use of ASP for developing conformant planners has been done in the past (see, for instance, [22] and [10]). In what follows, we outline an alternative approach that uses the language of Epistemic Specifications. A thorough study of this approach will be presented elsewhere.

By a (*conformant*) *planning problem* we mean a triple $\mathcal{P} = \langle \mathcal{D}, \Sigma, g \rangle$ which consists of a system description $\mathcal{D}$ of an action language $\mathcal{AL}$, a collection of the possible initial states Σ , and a goal g where g is a set of fluent literals. A sequence α of actions is called a *solution* to $\mathcal{P}$ if α is executable in every state of Σ and the execution of α in every such state makes the goal g true. (For more details, see [22].) For simplicity we assume that our system description $\mathcal{D}$ is deterministic, i.e. uncertainty is caused only by incompleteness of information about the initial state.

We will be interested in looking for the solutions of the planning problem that have exactly n steps with one action occurring at each step. Constant n is often referred to as the *horizon*. The proposed approach consists of the following steps:

1. A planning problem $\mathcal{P} = \langle \mathcal{D}, \Sigma, g \rangle$ is translated into planning problem $\mathcal{P}' = \langle \mathcal{D}', \Sigma', g' \rangle$ such that $\mathcal{D}'$ is a system description of $\mathcal{AL}$ that contains no impossibility condition and α is a solution of $\mathcal{P}$ iff it is a solution of $\mathcal{P}'$.
2. Planning problem $\mathcal{P}'$ is then translated into an epistemic logic program $\tau_n(\mathcal{P}')$ (where n is the horizon) such that the world views of $\tau_n(\mathcal{P}')$ correspond to the solutions of $\mathcal{P}'$ having length n .

³ The proofs for soundness and completeness of the algorithm are given in [15].

$\mathcal{D}'$ is obtained from $\mathcal{D}$ by expanding its signature by a new inertial fluent *is_impossible* and replacing every impossibility condition

$$\mathbf{impossible} \ a \ \mathbf{if} \ l_1, \dots, l_n$$

by causal law

$$a \ \mathbf{causes} \ is_impossible \ \mathbf{if} \ l_1, \dots, l_n.$$

The new collection Σ' of initial states and the new goal are defined as follows:

$$\begin{aligned} \Sigma' &=_{def} \{ \sigma' : \sigma' = \sigma \cup \{ \neg is_impossible \} \text{ for every } \sigma \in \Sigma \} \\ g' &=_{def} g \cup \{ \neg is_impossible \} \end{aligned}$$

The translation τ_n of $\mathcal{P}'$ into an epistemic logic program consists of several steps. We use f to denote a fluent atom, l a fluent literal, $body$ a comma-delimited list of fluent literals, and $h(body, S)$ the comma-delimited list of literals of the form $holds(f, S)$ or $\neg holds(f, S)$ for each l that is a member of $body$ where $l = f$ or $l = \neg f$, respectively. S ranges over natural numbers (often called *steps*) from 0 to the horizon n .

1. Causal laws of $\mathcal{D}'$ are translated into logic programming rules in the usual way.
 - (a) Each dynamic causal law of the form

$$a \ \mathbf{causes} \ l \ \mathbf{if} \ body$$

for which $l = f$ is translated into the rule

$$holds(f, S+1) \leftarrow occurs(a, S), h(body, S), S < n$$

and, where $l = \neg f$, into the rule

$$\neg holds(f, S+1) \leftarrow occurs(a, S), h(body, S), S < n.$$

- (b) Each static causal law (also referred to as a state constraint) of the form

$$l \ \mathbf{if} \ body$$

for which $l = f$ is translated into the rule

$$holds(f, S) \leftarrow h(body, S)$$

and, where $l = \neg f$, into the rule

$$\neg holds(f, S) \leftarrow h(body, S).$$

2. Inertia axioms and awareness axiom are also included in $\mathcal{D}'$:

$$\begin{aligned} holds(F, S+1) &\leftarrow holds(F, S), \text{ not } \neg holds(F, S+1), S < n. \\ \neg holds(F, S+1) &\leftarrow \neg holds(F, S), \text{ not } holds(F, S+1), S < n. \\ holds(F, 0) &\text{ or } \neg holds(F, 0). \end{aligned}$$

where F ranges over fluents.

3. Problem dependent relation $goal(S)$ is defined by the rule

$$goal(S) \leftarrow h(g'_1, \dots, g'_k, S).$$

where $g'_1, \dots, g'_k$ is the list of all elements of g' .

4. The initial situation is defined by the collection of disjunctions of the form

$$h_1 \text{ or } \dots \text{ or } h_m.$$

where each h is a literal of the form $holds(f, 0)$ or $\neg holds(f, 0)$.

5. A collection of rules, called the *epistemic conformant planning module*, is added:

$$\begin{aligned} occurs(A, S) &\leftarrow M occurs(A, S), S < n. \\ \neg occurs(A_2, S) &\leftarrow occurs(A_1, S), A_1 \neq A_2. \\ success &\leftarrow goal(n). \\ \leftarrow success, \text{ not } K success. \\ \leftarrow \text{ not } M success. \end{aligned}$$

Here each A ranges over actions. Intuitively, the first two rules form the *generate* part of the program. Note the fact that the first rule can be used as a generator due to the new semantics for modal operator M . (Recall *Example 3*.) Used together with the previously defined rules, these two rules allow for the generation of multiple world views. For every such world view W and every step s such that $0 \leq s < n$, there is exactly one action a such that an atom of the form $occurs(a, s)$ belongs to all belief sets of W .

The next rule defines the *success* of the planning process. Since we are only considering solutions of length n (i.e., the last action in the solution sequence occurs at step $n-1$), the *goal* must be met at step n .

The last two rules define constraints which must be satisfied by each world view of the program. The first says that it is impossible to achieve *success* in one belief set of a world view without achieving it in all its belief sets. The second guarantees existence of at least one belief set containing *success*. Together they ensure that *success* is achieved in all belief sets of a world view without otherwise eliminating individual belief sets.

This completes our construction of $\tau_n(\mathcal{P}')$.

Theorem 3. [Correctness of the Proposed Conformant Planning Algorithm]⁴

Given a conformant planning problem $\mathcal{P} = \langle \mathcal{D}, \Sigma, g \rangle$ where $\mathcal{D}$ is both deterministic and consistent (see [22]), the following is true with respect to the approach given above.

- For every world view W of $\tau_n(\mathcal{P}')$, belief sets of W coincide on atoms of the form $occurs(a, s)$.
- A sequence $\alpha = a_0, \dots, a_{n-1}$ of actions is a solution of $\mathcal{P}$ of length n iff there is a world view W of $\tau_n(\mathcal{P}')$ such that $occurs(a, k)$ belongs to its belief sets iff a is the k th element a_k of α .

We conclude by illustrating this algorithm with a simple example.

⁴ Proof of this theorem will be included in an expanded version of paper.

Example 5. [Yale Shooting Problem]

The agent is operating in a domain in which there is a turkey and a gun. The turkey can be alive or not. The gun may be loaded or not. If the gun is loaded and the trigger is pulled, then the turkey will be dead. Pulling the trigger will unload the gun. The agent can load the gun, but this action is impossible if the gun is already loaded. The domain can be represented by the following domain description $\mathcal{D}$:

pull_trigger causes $\neg$ alive if loaded
pull_trigger causes $\neg$ loaded
load causes loaded
impossible load if loaded

For our example, we give the following instance of the problem. Initially the turkey is alive, but it is unknown if the gun is loaded or not. The set of initial states is $\Sigma = \{\{alive, loaded\}, \{alive, \neg loaded\}\}$. The goal is to kill the turkey; i.e., $g = \{\neg alive\}$.

This planning problem $\mathcal{P} = \langle \mathcal{D}, \Sigma, g \rangle$ is translated into P' by replacing the impossibility condition by the causal law

load causes is_impossible if loaded

and adding *$\neg is_impossible$* to g and to each initial state in Σ .

The planning problem P' is then translated into the epistemic logic program $\tau(P')$ consisting of the following rules with variables ranging over appropriate domains:

```
% translated domain description
 $\neg holds(alive, S+1) \leftarrow occurs(pull\_trigger, S), holds(loaded, S), S < n.$ 
 $\neg holds(loaded, S+1) \leftarrow occurs(pull\_trigger, S), S < n.$ 
 $holds(loaded, S+1) \leftarrow occurs(load, S), S < n.$ 
 $holds(is\_impossible, S+1) \leftarrow occurs(load, S), holds(loaded, S), S < n.$ 
% inertia and awareness axioms
 $holds(F, S+1) \leftarrow holds(F, S), \text{not } \neg holds(F, S+1), S < n.$ 
 $\neg holds(F, S+1) \leftarrow \neg holds(F, S), \text{not } holds(F, S+1), S < n.$ 
 $holds(F, 0) \text{ or } \neg holds(F, 0).$ 
% initial condition
 $holds(alive, 0).$ 
 $\neg holds(is\_impossible, 0).$ 
% goal
 $goal(S) \leftarrow \neg holds(alive, S), \neg holds(is\_impossible, S).$ 
% conformant planning module
 $occurs(A, S) \leftarrow M occurs(A, S), S < n.$ 
 $\neg occurs(A_2, S) \leftarrow occurs(A_1, S), A_1 \neq A_2.$ 
 $success \leftarrow goal(n).$ 
 $\leftarrow success, \text{not } K success.$ 
 $\leftarrow \text{not } M success.$ 
```

If the horizon is set to $n=3$, the program has one world view. The world view has two belief sets: one in which the gun is initially loaded and one in which it isn't. Both, however, contain the same sequence of actions: $occurs(pull_trigger, 0)$, $occurs(load, 1)$, and $occurs(pull_trigger, 2)$. Regardless of the initial situation, the goal is reached at the horizon since $\neg holds(alive, 3)$ is in both belief sets and $\neg holds(is_impossible, 3)$ is in neither; hence, the sequence of actions is a conformant plan.

7 Related Work

In [24], Yan Zhang investigated the computational properties of epistemic logic programs, leading him to the development of an algorithm for computing world views. This effort culminated in the development of a solver known as `Wviews`[23] implemented by Michael Kelly. `Wviews` is based on ES1994 and relies on the ASP solver `DLV`[9].

In [21], Truszczyński offered alternative semantics to ES1991 by using Kleene’s three-valued logic[16]. It was not Truszczyński’s goal to address the issue of unintended world views; nonetheless, his added requirement that a world view be *non-empty* eliminated one form of unintended world view, whereas his additional requirement that belief sets of a world view be *consistent* put the language more in sync with modern ASP semantics, eliminating the world view containing the set of all literals. Later in the paper, a two-valued logic was used to provide a semantic framework in which three different languages based on Epistemic Specifications were defined. Truszczyński went on to derive complexity results for Epistemic Specifications that suggest the language is strictly more expressive than ASP. Finally, he elaborated on the use of the language for modeling various problems that require meta-reasoning.

In [11], Faber and Woltran presented a framework for combining ASP programs in a manner that allows consequences of programs to be used as input to other programs. They also presented a technique for compiling such a framework into a single ASP program using the concept of manifold programs. A manifold program is an expanded encoding of an ASP program that allows for reasoning about the consequences of its answer sets from within the encoding, thus providing a limited capacity for introspective reasoning similar to that provided by modal operators in the language of Epistemic Specifications. Their approach relies on the ASP extension known as *weak constraints*[7]. For certain epistemic logic programs, the methods described by Faber and Woltran may provide an efficient method for finding world views.

Gelfond’s 2011 paper[14] was the latest publication proposing an update to the semantics of Epistemic Specifications, ES2011, with a stated goal of eliminating unintended world views. His new reduct solved the problem of unintended world views associated with *K*; however, the problem of unintended world views associated with recursion through *M* remained. Our paper can be considered a continuation of that work.

Cui, Z. Zhang and Zhao[8] investigated the problem of grounding an epistemic logic program. That work culminated in the development of a grounder known as `ESParser`. Their more recent efforts[25] include the development of an associated solver called `ESmodels`[20] that relies on ASP solver `claspD`[18]. Their work is based on ES2011. The algorithm given for `ESmodels` uses an approach that is, at its core, much like the one used in the algorithm presented in this paper. They also include an optimization they refer to as the “EFFICIENT ESMODELS” algorithm that may suggest an approach that is applicable to our algorithm as well.

Balai and Kahl[5] extend the language of Epistemic Specifications by providing a framework for explicitly specifying a sorted signature in the same manner as that done for ASP with `SPARC`[3,4,2]. They call the language *Epistemic Logic Programs with Sorts (ELPS)*. It is based on the syntax and semantics of ES2014 as described herein. An ELPS solver[1] was implemented by Balai using a modification of Algorithm 1.

8 Conclusion

The version of Epistemic Specifications defined herein provides a reasonable solution to the problem of unintended world views that had remained unresolved for the past two decades. The new semantics eliminates unintended world views stemming from recursion through modal operators, while the requirement that world views be non-empty and contain only consistent sets eliminates still other sources of unintended world views and puts Epistemic Specifications in sync with current ASP semantics. We believe these changes represent a significant improvement to the language, making it arguably closer to the intuitive meaning, and thus better suited for knowledge representation and reasoning, and the design of intelligent agents.

The algorithm will hopefully lead to an efficient implementation of an epistemic logic program solver that will allow the language to be of more practical use. Applications such as the conformant planner show the potential of Epistemic Specifications for modeling complex tasks. Such applications also show that programs with multiple world views can be useful.

9 Future Work

Future work may include improving the algorithm presented herein to create a more efficient epistemic logic program solver. A proof-of-concept solver has been developed as a loosely-coupled system using Potassco's `clingo`[18]. It is hoped that a more tightly-coupled system with improvements to the algorithm will be developed.

Once a reliable and reasonably efficient implementation exists, our next goal will be to devise applications and methodologies for solving problems for which the language of Epistemic Specifications seems ideally suited. The proposed framework for conformant planning is one such application.

Finally, it will also be important to look at the mathematical properties of epistemic logic programs.

Acknowledgments

Work by Gelfond and Zhang was partially supported by NFS grants IIS-1018031 and CNS-1359359.

References

1. Balai, E.: ELPS, Texas Tech University, software & documentation available for download at <https://github.com/iensen/elps/wiki/>
2. Balai, E.: SPARC, Texas Tech University, software & documentation available for download at <https://github.com/iensen/sparc/wiki/>
3. Balai, E., Gelfond, M., Zhang, Y.: *SPARC*–Sorted ASP with consistency restoring rules. CoRR abs/1301.1386 (2013)
4. Balai, E., Gelfond, M., Zhang, Y.: Towards answer set programming with sorts. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. Lecture Notes in Artificial Intelligence, vol. 8148. Springer-Verlag (2013)

5. Balai, E., Kahl, P.: Epistemic logic programs with sorts. In: Inclezan, D., Maratea, M. (eds.) ASPOCP 2014. CoRR (2014)
6. Balduccini, M., Son, T.C. (eds.): Logic Programming, Knowledge Representation, and Non-monotonic Reasoning—Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday, Lecture Notes in Computer Science, vol. 6565. Springer (2011)
7. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. *IEEE Transactions on Knowledge and Data Engineering* 12(5), 845–860 (2000)
8. Cui, R., Zhang, Z., Zhao, K.: ESParser: An epistemic specification grounder. In: Delgrande, J.P., Faber, W. (eds.) CSSS-12. pp. 1823–1827. IEEE Computer Society CPS (2012)
9. DLVSYSTEM s.r.l.: DLV (2012), software & documentation available for download at <http://www.dlvsystem.com/>
10. Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: A logic programming approach to knowledge-state planning, II: The DLV^K system. *Artificial Intelligence* 144(1–2), 157–211 (2003)
11. Faber, W., Woltran, S.: Manifold answer-set programs and their applications. In: Balduccini and Son [6], pp. 44–63
12. Gelfond, M.: Strong introspection. In: Dean, T.L., McKeown, K. (eds.) AAAI-91. vol. 1, pp. 386–391. AAAI Press / The MIT Press (1991)
13. Gelfond, M.: Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence* 12(1–2), 89–116 (1994)
14. Gelfond, M.: New definition of epistemic specifications. In: Delgrande, J.P., Faber, W. (eds.) LPNMR-11. Lecture Notes in Computer Science, vol. 6645, pp. 260–265. Springer (2011)
15. Kahl, P.: Refining the Semantics for Epistemic Logic Programs. PhD dissertation, Texas Tech University, Lubbock, TX, USA (May 2014)
16. Kleene, S.C.: Introduction to Metamathematics. University Series in Higher Mathematics, D. Van Nostrand, Princeton, NJ (1952)
17. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25, 369–389 (1999)
18. Potassco: clasp, claspD, clingo, gringo (2014), University of Potsdam, software & documentation available for download at <http://potassco.sourceforge.net/>
19. Smith, D.E., Weld, D.S.: Conformant graphplan. In: Proceedings of the 15th National Conference on Artificial Intelligence/10th Conference on Innovative Applications of Artificial Intelligence. pp. 889–896. AAAI-98/IAAI-98, AAAI (1998)
20. Southeast University: ESparser, ESmodels (2013), software & documentation available for download at http://cse.seu.edu.cn/people/seu_zzz/index.htm
21. Truszczyński, M.: Revisiting epistemic specifications. In: Balduccini and Son [6], pp. 315–333
22. Tu, P.H., Son, T.C., Gelfond, M., Morales, A.R.: Approximation of action theories and its application to conformant planning. *Artificial Intelligence* 175(1), 79–119 (January 2011)
23. University of Western Sydney AI Research Group: Wviews (2007), UWS AI Research Group (Intelligent Systems Laboratory), software & documentation available for download at <http://staff.scm.uws.edu.au/%7eyan/Wviews.html>
24. Zhang, Y.: Computational properties of epistemic logic programs. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) KR-06. pp. 308–317. AAAI Press (2006)
25. Zhang, Z., Zhao, K., Cui, R.: ESmodels: An inference engine of epistemic specifications. In: Luo, J. (ed.) ICTAI 2013. pp. 769–774. IEEE (2013)