Epistemic Logic Programs with World View Constraints

Patrick Kahl and Anthony Leclerc

June 4, 2018

Epistemic Logic Programs with World View Constraints

Patrick Thor Kahl
Space and Naval Warfare Systems Center Atlantic
North Charleston, SC, USA
patrick.kahl@navy.mil

Anthony P. Leclerc
Space and Naval Warfare Systems Center Atlantic | College of Charleston
North Charleston, SC, USA | Charleston, SC, USA
anthony.leclerc@navy.mil | leclerca@cofc.edu

Abstract

An epistemic logic program is a set of rules written in the language of Epistemic Specifications, an extension of the language of answer set programming that provides for more powerful introspective reasoning through the use of modal operators K and M. We propose adding a new construct to Epistemic Specifications called a *world view constraint* that provides a universal device for expressing global constraints in the various versions of the language. We further propose the use of subjective literals (literals preceded by K or M) in rule heads as syntactic sugar for world view constraints. Additionally, we provide an algorithm for finding the world views of such programs.

2012 ACM Subject Classification Software and its engineering → Software notations and tools → General programming languages → Language features → Constraints

Keywords and phrases Epistemic Specifications, Epistemic Logic Programs, Constraints, World View Constraints, World View Rules, WV Facts, Answer Set Programming, Logic Programming

Acknowledgements The authors wish to express their thanks to Evan Austin, Michael Gelfond, and ICLP anonymous reviewers for their valued suggestions and comments on drafts of this work.

1 Introduction

The language of *Epistemic Specifications* extends answer set programming (ASP) by adding modal operators K ("known") and M ("may be true"). It was introduced by Gelfond [16] after observing a need for more powerful introspective reasoning than that offered by ASP alone. A program written in this language is called an *epistemic logic program* (ELP), with semantics defined using the notion of a *world view*—a collection of sets of literals (*belief sets*), analogous to answer sets of an ASP program. Recent interest has led to a succession of proposed semantics [18, 22, 13, 36, 45] advocating differing perspectives with respect to the meaning of connectives and intended world views of programs. This clash of intuition is only one aspect of the problem as defining a semantics that facilitates understanding and yet accurately reflects intuition appears to be quite difficult as discussed in Section 2.

In this paper, we don't try to resolve the clash; instead, we focus on the important problem of modeling knowledge using purely epistemic constraints. With the original semantics, such constraints could be used to eliminate possible worlds. As will be shown, this property was lost with the more recent semantics. This leads to substantial difficulties in modeling knowledge. Thus, in an attempt to facilitate ELP development in the midst of language evolution, we propose extending the language with a syntactic construct called a *world view* *constraint* (WVC) to distinguish certain constraints as global. WVCs are universal—immune by design to the various devices (e.g., maximality requirements) used to tweak the semantics.

As an introductory example, let us look at a simple epistemic logic program that features a purely epistemic constraint:

p `or` q.
$\leftarrow$ `not` $\mathrm{K}\, p$. % purely epistemic constraint

The second rule is purely epistemic in that its body consists solely of a subjective literal whose interpretation is global in the sense that its truth value depends on the entire collection of belief sets in some possible world view rather than some current (local) working set. To be precise, the truth value of `not` $\mathrm{K}\, p$ depends on whether p is in all belief sets of some possible world view under consideration. For example, considering $W = \{\{p\}, \{q\}\}$ as a possible world view, `not` $\mathrm{K}\ p$ evaluates to *true*, thus violating the constraint.

As W is the only possible world view that is consistent with the rest of the program (i.e., the first rule), this program has no world view under the original semantics [16]. However, under most of the recently proposed semantics [22, 13, 36], its world view is $\{\{p\}\}$—a result that some may consider unexpected. To achieve the same result as that of the original semantics, we propose replacing the second rule with the following:

$\overset{\texttt{wv}}{\leftarrow}$ `not` $\mathrm{K}\, p$. % world view constraint

which results in no world view for any of the proposed ELP semantics (if extended with our new construct). This motivating example and others are discussed in Sections 3 and 5.

The paper is organized as follows. We begin with a summary of related work in development of the language semantics and ELP solvers. Next we discuss the use of constraints in both ASP and Epistemic Specifications, providing motivational argument for the introduction of WVCs. We then present the syntax and semantics of the extended language. We follow with examples demonstrating its use. Finally, we give an algorithm for computing the world views of an ELP with WVCs, and close with suggestions for related extensions.

2 Background and Related Work

With his good friend and colleague Vladimir Lifschitz, the foundations for what we now call *answer set programming (ASP)* had been laid down in the seminal works of Michael Gelfond [20, 21] by 1991. It seems strange in hindsight that, in the same year, a far less known language called *Epistemic Specifications* was proposed by Gelfond [16] in an attempt to address an observed inadequacy in the expressiveness of its better known predecessor. Gelfond noticed that the following ASP program does not entail a required interview for a scholarship applicant whose eligibility is not able to be established:

% rules for scholarship eligibility at a certain college where S represents a scholarship applicant
$eligible(S) \leftarrow highGPA(S).$
$eligible(S) \leftarrow fairGPA(S),\ minority(S).$
$\neg eligible(S) \leftarrow \neg highGPA(S),\ \neg fairGPA(S).$
% ASP attempt to express that an interview is required if applicant eligibility can't be determined
$interview(S) \leftarrow$ `not` $eligible(S)$, `not` $\neg eligible(S)$.
% applicant data
$fairGPA(mike)$ `or` $highGPA(mike)$.

The program correctly reflects that Mike's eligibility can not be determined, but its answer sets, $\{fairGPA(mike), interview(mike)\}$ and $\{highGPA(mike), eligible(mike)\}$, do not conclude that an interview is required since only one contains $interview(mike)$.

Gelfond's solution was to extend the language by adding modal operator K ("known") and changing the fourth rule above as follows:

% updated rule to express interview requirement using modal operator K

$interview(S) \leftarrow$ `not` K $eligible(S)$, `not` K $\neg eligible(S)$.

The updated rule says that $interview(S)$ is to be believed if both $eligible(S)$ and $\neg eligible(S)$ are each *not known* (i.e., not in all belief sets of the world view). The program has world view $\{\{fairGPA(mike), interview(mike)\}, \{highGPA(mike), eligible(mike), interview(mike)\}\}$ with its belief sets both containing $interview(mike)$; thus, the required interview is entailed.

Although the language of Epistemic Specifications was revised in the first years of its introduction, after 1994 [6, 17] (referred to hereafter as *ES1994*) little concerning its semantics was seen in the literature for almost two decades. In the intervening years before 2011, Chen [11] proposed *GOL*, a generalization of Levesque's *logic of only knowing* ($\mathcal{OL}$) [29], that "covers Gelfond's important notion of Epistemic Specifications." Preda [34] proposed an alternative to Epistemic Specifications using multiple levels of negation (perhaps a precursor to the 2016 Shen-Eiter proposal discussed later). Wang and Yan Zhang [42] offered another alternative, proposing an epistemic extension to Pearce's *equilibrium logic of here-and-there* [33]. Efforts in 2011 by Faber & Woltran [14] and Truszczyński [40] mark the beginning of a renewed interest in Epistemic Specifications.

With the resurgence of interest, Gelfond felt an update to Epistemic Specifications was needed. His proposal [18] (referred to hereafter as *ES2011*) specifically addressed unintended world views due to recursion through modal operator K, as exemplified here:

$p \leftarrow$ K p.

Under ES1994 semantics, this program has two world views, $\{\{\}\}$ and $\{\{p\}\}$. Under ES2011 semantics, only the first is a world view, which is arguably more intuitive. It was observed, however, that unintended world views due to recursion through modal operator M remain, as demonstrated by the following one-line program:

$p \leftarrow$ M p.

Under ES2011 semantics the program has two world views, $\{\{\}\}$ and $\{\{p\}\}$. This result did not seem intuitive. Following Gelfond's lead, Kahl et al. [23, 22] proposed another update (referred to hereafter as *ES2014*) to address the issue, with semantics supporting only the latter world view.

It was suggested by Fariñas del Cerro et al. [13] that there remain unintended world views for certain programs with ES2014 semantics, particularly the following:

$p \leftarrow$ M q, `not` q.
$q \leftarrow$ M p, `not` p.

Per ES2014, the program has two world views, $\{\{\}\}$ and $\{\{p\}, \{q\}\}$, of which the first, they argue, seems unintended. Their notion of *autoepistemic equilibrium models* (AEEMs) attempts to address this concern with a new epistemic extension of equilibrium logic that includes a maximality condition on epistemic equilibrium models. Using AEEMs successfully eliminates $\{\{\}\}$ from the above program's world views.

Shen and Eiter [36] offered another update to the semantics, albeit using different syntactic notation, that focused on resolving unintended world views due to:

- *epistemic circular justification* in which a literal is considered *true* solely on the assumption that it is in all belief sets (i.e., belief in ℓ is justified only by K ℓ); and
- not satisfying the *property of knowledge minimization with epistemic negation.*

The *property of knowledge minimization with epistemic negation* is based on a maximality requirement on a *guess* (i.e., a set of *epistemic negations*—equivalent to subjective literals of the forms `not` K ℓ and M ℓ—considered *true* within the program under consideration) for its associated collection of belief sets to be a world view, all other conditions being satisfied. In [24], the authors provided a revision of ES2014 semantics by adding this maximality requirement (referred to hereafter as *ES2016*).

Following suit, Zhizheng Zhang [46] updated his semantics for *answer set programming with graded modality* (ASP$^{\mathsf{GM}}$) by adding a maximality condition in line with Shen and Eiter. With some syntactic liberty, Epistemic Specifications can be viewed as a proper subset of ASP$^{\mathsf{GM}}$ with ASP$^{\mathsf{GM}}$ allowing for expressing a lower and upper bound on the number of belief sets containing a specified literal within a world view. (We will revisit ASP$^{\mathsf{GM}}$ in Section 7.)

Recently, Yan Zhang and Yuanlin Zhang [45] offered a different semantics for ELPs, with a stricter view on *circular justification.* To illustrate, they argue that the program

$p \leftarrow \mathrm{M}\, p.$

should have the world view $\{\{\}\}$ rather than $\{\{p\}\}$ as they do not consider circular justification of p as sufficient reason to accept the latter. To them, justification for $\mathrm{M}\, p$ being *true* requires that *belief in p is forced in some belief set of the rational agent.*[1] Others argue that $\mathrm{M}\ p$ is equivalent to `not` K `not` p and that the *rationality principle* (which states that a rational agent should believe only what it is forced to believe) favors *not knowing* (`not` K) over *knowing* (K), so $\{\{p\}\}$ is the preferred world view. In contrast, Zhang & Zhang use this same principle to argue against $\{\{p\}\}$ since the *possibility* of p is not viewed as enough by itself to force belief in p.

Regardless of differing views, it appears there remains room for improvement. As one example, in [36] the *problem of unintended world views due to recursion through* M is defined as a semantics for which "its world views do not satisfy the *property of knowledge minimization with epistemic negation.*" Use of this definition avoids the question of whether, based on intuition, a program has unintended world views. Consider again the program

$p \leftarrow \mathrm{M}\ q,\ \texttt{not}\ q.$
$q \leftarrow \mathrm{M}\ p,\ \texttt{not}\ p.$

for which $\{\{p\}, \{q\}\}$ is the only world view per this knowledge minimization property. Adding

$r \leftarrow \mathrm{M}\ p,\ \mathrm{M}\ q.$

results in the world view $\{\{p, r\}, \{q, r\}\}$, as one might expect. But now if we add the rule

$s \leftarrow \mathrm{K}\ r.$

we get *two* world views: $\{\{p, r, s\}, \{q, r, s\}\}$ and $\{\{\}\}$. In lieu of the other results, this seems unintuitive in spite of following the *property of knowledge minimization with epistemic negation.* We believe this demonstrates the difficulty in defining an intuitive semantics.

In conjunction with development of the language, there has been concomitant development of tools for finding world views. Attempts at developing a solver or inference engine include `ELMO` by Watson [43], `sismodels` by Balduccini [3], `Wviews` by Kelly [25, 26, 41] using Yan Zhang's algorithm [44], `ESmodels` by Zhizheng Zhang et al. [35, 47], `ELPS` by Balai [1, 2], `ELPsolve` by the authors [24], `EP-ASP` by Son et al. [27, 37], `EHEX` by Strasser [38], and `selp` by Bichler et al. [7, 8]. A thorough discussion of these tools is left for another paper [28]. It deserves note, however, that all extant solvers use an ASP solver for backend processing, and as ASP solver development has matured, ELP solver development has slowly followed.

3 Motivation for World View Constraints

It is well known (see, for example, Proposition 2 in [31]) that constraints (headless rules) in an ASP program have the net effect of, at most, ruling out certain answer sets from the

[1] See the notion of an *externally-supported M-cycle* in [23].

program (modulo its constraints). To illustrate, consider the following ASP program:

p `or` q.
$p \leftarrow q$.

which has one answer set $\{p\}$. If we add the constraint

$\leftarrow p,$ `not` q.

the resulting program has no answer set since $\{p\}$ violates this constraint.

With Epistemic Specifications, constraints can have an additive or subtractive effect on belief sets or entire world views. Consider, for example, the following ELP:

p `or` q.
$r \leftarrow \mathrm{M}\ q$.

with world view $\{\{p, r\}, \{q, r\}\}$. If we add the constraint

$\leftarrow q$.

the resulting program has world view $\{\{p\}\}$. Let's look at another example:

p `or` q.
$r \leftarrow \mathrm{M}\ p$.
s `or` $t \leftarrow \mathrm{K}\ p$.

This program has a single world view, $\{\{p, r\}, \{q, r\}\}$. If we add the constraint

$\leftarrow \mathrm{M}\ p,\ \mathrm{M}\ q$.

the resulting program has two world views per ES2016: $\{\{p, r, s\}, \{p, r, t\}\}$ and $\{\{q\}\}$.

The previous examples illustrate potential differences in the effect of constraints on an ELP compared to an ASP program. The last may also show how constraints can be a possible source of confusion with respect to world views.[2] Consider another example:

p `or` q.
$\leftarrow$ `not` $\mathrm{K}\ p$.

Under ES2016 semantics, its world view is $\{\{p\}\}$; however, under the original semantics [16], the program has no world view. This raises the question:

Which result is intended?

If the intent of the constraint is to rule out *world views* that do not contain p in every belief set, then the latter (from the original semantics) would seem correct. Under the later semantics, the net effect of the constraint is to eliminate *belief sets* that would otherwise result in a world view that violates the constraint.

For ES2014 semantics, it was shown in [22] that, in general, to eliminate *world views* that do not contain p in every belief set (and not simply eliminate belief sets from a world view that would otherwise not meet this requirement), two constraints are required instead of the one given above, resulting here in the following program:

p `or` q.
$\leftarrow p,$ `not` $\mathrm{K}\ p$.
$\leftarrow$ `not` $\mathrm{M}\ p$.

So with ES2014 semantics we now have a program with no world view. The same is true in this case with the later ES2016 semantics; however, the new maximality requirement in ES2016 means such "tricks" won't work for all programs. Consider the following:

$p \leftarrow \mathrm{M}\ q,$ `not` q.
$q \leftarrow \mathrm{M}\ p,$ `not` p.
$r \leftarrow \mathrm{M}\ p,\ \mathrm{M}\ q$.

Under ES2014 semantics, $\{\{\}\}$ and $\{\{p, r\}, \{q, r\}\}$ are the world views. Per ES2016, only the latter is a world view. If we now add the constraint

[2] Should the program even have a world view? Under the earlier ES1994 semantics, it does not!

$\leftarrow \mathrm{K}\, r.$

the resulting program has world view $\{\{\}\}$, which is not an ES2016 world view without this constraint. We observe that there does not appear to be a general way to simply rule out world views under ES2016 semantics. This observation leads to our thesis.

As the semantics of Epistemic Specifications has evolved to address unintended world views and support intuition with respect to certain programs, we believe the added complexity has had a negative side effect with respect to intuitive understanding of certain other programs—particularly those involving constraints with subjective literals. Thus, in an attempt to facilitate correct problem encoding/program development in line with intuition, we propose a new language construct called a *world view constraint* (WVC) and introduce symbol $\overset{\mathrm{wv}}{\Leftarrow}$ read as "it is not a world view (if...)" for use in forming a WVC. For example,

$\overset{\mathrm{wv}}{\Leftarrow}\ \mathrm{K}\, p.$

is read "it is not a world view if p is known" and means (informally) that any world view satisfying $\mathrm{K}\, p$ is ruled out from the set of world views of the program under consideration. This is analogous to how constraints affect answer sets in ASP, though at the world view level for Epistemic Specifications.

4 Syntax and Semantics

For the purpose of demonstrating the use of WVCs, we first define the syntax and semantics for two versions of the language: ES2014 and ES2016. We direct the reader to the papers referenced earlier for information on other versions of Epistemic Specifications. We present our proposal for extending the language in Section 4.3, and follow by suggesting a means of expressing the bounds for the grounding of variables within the context of the new constructs.

In general, the syntax and semantics of the language of Epistemic Specifications follow that of ASP with the notable addition of modal operators K and M, plus the new notion of a *world view* which is a collection of *belief sets* analogous to answer sets. We assume familiarity with ASP [5, 9, 15, 19, 30]. We use $\mathrm{AS}(\mathcal{P})$ to denote the set of answer sets of ASP program $\mathcal{P}$. We use symbol $\models$ for *satisfies* and $\not\models$ for *does not satisfy*.

4.1 Syntax [ES2014 and ES2016]

An epistemic logic program is a set of rules of the form

$$\ell_1 \ \texttt{or}\ \ldots\ \texttt{or}\ \ \ell_k \leftarrow e_1,\ \ldots,\ e_n.$$

where $k \geq 0$, $n \geq 0$, each ℓ_i is a *literal* (an atom or a classically-/strongly-negated atom; called an *objective literal* when needed to avoid ambiguity), and each e_i is a literal or a *subjective literal* (a literal immediately preceded by K or M) possibly preceded by `not` (default negation). As in ASP, a rule having an objective/subjective literal with a variable term is a shorthand for all ground instantiations of the rule. By *body*(R) we denote the set $\{e_1, \ldots, e_n\}$ from the body of rule R.

4.2 Semantics

▶ **Definition 1.** [When a Subjective Literal Is Satisfied]
Let W be a non-empty set of consistent sets of ground literals, and ℓ be a ground literal.

- $W \models \mathrm{K}\,\ell$ if $\forall A \in W:\ \ell \in A.$
- $W \models \mathrm{M}\,\ell$ if $\exists A \in W:\ \ell \in A.$
- $W \models \texttt{not}\ \mathrm{K}\,\ell$ if $\exists A \in W:\ \ell \notin A.$
- $W \models \texttt{not}\ \mathrm{M}\,\ell$ if $\forall A \in W:\ \ell \notin A.$

▶ **Definition 2.** [Modal Reduct]
Let Π be a ground epistemic logic program, W be a non-empty set of consistent sets of ground literals, and ℓ be a ground literal. We denote by Π^W the *modal reduct of* Π *with respect to* W defined as the ASP program[3] obtained from Π by replacing/removing subjective literals in rule bodies or deleting associated rules per the following table:

subjective literal φ	**if** $W \models \varphi$ **then...**	**if** $W \not\models \varphi$ **then...**
$\mathrm{K}\,\ell$	replace $\mathrm{K}\,\ell$ with ℓ	delete rule containing $\mathrm{K}\,\ell$
$\texttt{not}\ \mathrm{K}\,\ell$	remove $\texttt{not}\ \mathrm{K}\,\ell$	replace $\texttt{not}\ \mathrm{K}\,\ell$ with $\texttt{not}\ \ell$
$\mathrm{M}\,\ell$	remove $\mathrm{M}\,\ell$	replace $\mathrm{M}\,\ell$ with $\texttt{not}\ \texttt{not}\ \ell$
$\texttt{not}\ \mathrm{M}\,\ell$	replace $\texttt{not}\ \mathrm{M}\,\ell$ with $\texttt{not}\ \ell$	delete rule containing $\texttt{not}\ \mathrm{M}\,\ell$

▶ **Definition 3.** [World View under ES2014 Semantics]
Let Π be a ground epistemic logic program and W be a non-empty set of consistent sets of literals. W is a *world view* of Π under ES2014 semantics if $W = \mathrm{AS}(\Pi^W)$.

▶ **Definition 4.** [Epistemic Negations[4]]
Let Π be a ground epistemic logic program, W be a non-empty set of consistent sets of literals, and ℓ be a ground literal. We denote by $E_P(\Pi)$ the set of distinct subjective literals appearing (regardless of being negated) in Π, each taking the form of $\texttt{not}\ \mathrm{K}\ell$ or $\mathrm{M}\ell$ (referred to as *epistemic negations*) as follows:

$$E_P(\Pi) = \{\ \texttt{not}\ \mathrm{K}\ell : \mathrm{K}\ell \text{ appears in } \Pi\ \} \cup \{\ \mathrm{M}\ell : \mathrm{M}\ell \text{ appears in } \Pi\ \}.$$

In context with Π, we use Φ to denote a subset of $E_P(\Pi)$, and denote by Φ_W the subset of epistemic negations in $E_P(\Pi)$ that are satisfied by W; i.e., $\Phi_W = \{\ \varphi : \varphi \in E_P(\Pi) \wedge W \models \varphi\ \}$.

▶ **Definition 5.** [World View under ES2016 Semantics]
Let Π be a ground epistemic logic program and W be a non-empty set of consistent sets of literals. W is a *world view* of Π under ES2016 semantics if:
(1) $W = \mathrm{AS}(\Pi^W)$; and (2) there is no W' such that $W' = \mathrm{AS}(\Pi^{W'})$ and $\Phi_{W'} \supset \Phi_W$.[5]

4.3 World View Constraints and World View Rules

We extend the language of Epistemic Specifications by introducing a *world view constraint* as a construct for restricting the world views of an ELP, and a *world view rule* as a syntactic device for specifying a world view constraint in an effort to facilitate problem encoding/program development. The syntax and semantics of ES2016 are assumed here for the core ELP, though the definitions should work with other language versions.

4.3.1 World View Constraints

A *world view constraint* (WVC) is an epistemic logic program rule of the form

$$\stackrel{\texttt{wv}}{\leftarrow}\ s_1,\ \ldots,\ s_n.$$

where each s_i is a (possibly negated) subjective literal.[6]

[3] with nested expressions of the form $\texttt{not}\ \texttt{not}\ \ell$ as defined in [31]
[4] introduced in [36] using a different syntax
[5] The *maximality requirement* on Φ_W comes from the *general epistemic semantics* of Shen and Eiter [36].
[6] A negated subjective literal is of the form $\texttt{not}\ \mathrm{K}\ \ell$ or the form $\texttt{not}\ \mathrm{M}\ \ell$ in ES2016 syntax.

▶ **Definition 6.** [When a World View Constraint Is Violated]
Let W be a non-empty set of consistent sets of ground literals, and C be a ground WVC of the form $\overset{\texttt{wv}}{\leftarrow} s_1,\ \ldots,\ s_n$. We say that W *violates* C if $\forall s_i \in \mathit{body}(C) : W \models s_i$.[7]

▶ **Definition 7.** [Semantics of an ELP with WVCs]
Let Π be a ground ELP with WVCs such that $\Pi = \Pi_0 \cup \Pi_{\texttt{wvc}}$ where $\Pi_{\texttt{wvc}}$ is the set of all WVCs in Π and $\Pi_0 = \Pi \setminus \Pi_{\texttt{wvc}}$ (i.e., the part of the program without WVCs). Let W be a non-empty set of consistent sets of ground literals. W is a *world view* of Π if:
(1) W is a world view of Π_0; and (2) W does not violate any rule in $\Pi_{\texttt{wvc}}$.

Returning to our example, let Π be the following program, partitioned as shown:

$$\left.\begin{array}{l} p \leftarrow \mathrm{M}\ q,\ \texttt{not}\ q. \\ q \leftarrow \mathrm{M}\ p,\ \texttt{not}\ p. \\ r \leftarrow \mathrm{M}\ p,\ \mathrm{M}\ q. \end{array}\right\}\Pi_0$$
$$\overset{\texttt{wv}}{\leftarrow}\ \mathrm{K}\ r. \qquad \}\ \Pi_{\texttt{wvc}}$$

Per ES2016 semantics, Π_0 has one world view $W = \{\{p, r\}, \{q, r\}\}$, but by our definition W violates the WVC in $\Pi_{\texttt{wvc}}$ since $W \models \mathrm{K}\ r$; hence, Π has no world view.

4.3.2 World View Rules and World View Facts

A *world view rule* (WVR) is an epistemic logic program rule of the form

$$s_1\ \ \texttt{or}\ \ \ldots\ \ \texttt{or}\ \ s_k\ \overset{\texttt{wv}}{\leftarrow}\ s_{k+1},\ \ldots,\ s_n.$$

where each s_i is a (possibly negated) subjective literal. We define a WVR as follows:

$$s_1\ \ \texttt{or}\ \ \ldots\ \ \texttt{or}\ \ s_k\ \overset{\texttt{wv}}{\leftarrow}\ s_{k+1},\ \ldots,\ s_n.\ \ \overset{\text{def}}{=}\ \ \overset{\texttt{wv}}{\leftarrow}\ \texttt{not}\ s_1,\ \ldots,\ \texttt{not}\ s_k,\ s_{k+1},\ \ldots,\ s_n.$$

where $\texttt{not}\ \texttt{not}\ \varphi \equiv \varphi$ for a subjective literal φ. A WVR is thus syntactic sugar for a WVC.

Similar to a *fact* in ASP, the $\overset{\texttt{wv}}{\leftarrow}$ symbol can be omitted from a WVR with no body. We refer to such rules as *world view facts*, or *WV facts*,[8] and use below in our example:

$p \leftarrow \mathrm{M}\ q,\ \texttt{not}\ q.$
$q \leftarrow \mathrm{M}\ p,\ \texttt{not}\ p.$
$r \leftarrow \mathrm{M}\ p,\ \mathrm{M}\ q.$
$\texttt{not}\ \mathrm{K}\ r.$ % equivalent to $\overset{\texttt{wv}}{\leftarrow}\ \mathrm{K}\ r.$

Note that with these definitions, any WVC can be written as a WVR, or equivalently as a WV fact. To demonstrate, the following three rules are all strongly equivalent:

$\overset{\texttt{wv}}{\leftarrow}\ \mathrm{K}\ p,\ \texttt{not}\ \mathrm{K}\ q,\ \mathrm{M}\ r,\ \texttt{not}\ \mathrm{M}\ s.$ % expressed here as a WVC
$\texttt{not}\ \mathrm{K}\ p\ \ \texttt{or}\ \ \mathrm{K}\ q\ \overset{\texttt{wv}}{\leftarrow}\ \mathrm{M}\ r,\ \texttt{not}\ \mathrm{M}\ s.$ % expressed here as a WVR
$\texttt{not}\ \mathrm{K}\ p\ \ \texttt{or}\ \ \mathrm{K}\ q\ \ \texttt{or}\ \ \texttt{not}\ \mathrm{M}\ r\ \ \texttt{or}\ \ \mathrm{M}\ s.$ % expressed here as a WV fact

4.4 Grounding Concerns

The issue of grounding an ELP received attention by both Kelly [25] and Cui et al. [12]. In [23], Kahl proposed an ELP solver algorithm that first creates a corresponding ASP program from the ungrounded ELP, and then uses an ASP grounder to determine the associated ground terms. This requires the rules in the ELP to be *safe* in the sense that any variable

---

[7] Likewise, we say that W *satisfies* C (i.e., $W \models C$) if $\exists s_i \in \mathit{body}(C) : W \not\models s_i$.
[8] In addition to being a notational convenience, solver developers can avoid introducing a new token for the $\overset{\texttt{wv}}{\leftarrow}$ symbol since any WVC can be expressed as a (possibly disjunctive) WV fact.

term appearing in a rule has a corresponding *positive literal* (either an objective literal or a subjective literal of the form K ℓ) in the body with the same variable term.

Having only subjective literals of the form K ℓ available for rule safety is too restrictive for WVCs. One could argue that the use of a *sorted signature*, such as in an *epistemic logic program with sorts* [2], would suffice if rule safety were the only issue; however, being able to limit the grounding of variable terms to less than the full range of their acceptable domains is key to abstraction. Without such capability, flexibility and elaboration tolerance suffer.

To address the practical need of having a reasonable way to express limits on the domain of a variable term in a WVC, we propose an extended syntax for a WV fact as follows:

$$s_1 \ \texttt{or} \ \ldots \ \texttt{or} \ s_m \leftarrow d_1, \ \ldots, \ d_n.$$

where each s_i is a (possibly negated) subjective literal, and each d_i is a *domain atom*[9]—also referred to as a *domain predicate* [39]—or a *comparison atom* (typically expressed using an infix "built-in" predicate; e.g., $X \neq a$). The body is used here only to determine the appropriate grounding of variable terms in the head of the rule. The use of the $\leftarrow$ symbol is intentional as the body is not (after grounding and translation) part of any WVC.[10] The program rules below demonstrate the use of this extended syntax:

% domain atoms

$d\_x(a)$. $d\_x(b)$.

$d\_y(0)$. $d\_y(1)$. $d\_y(2)$. $d\_y(3)$.

% WV fact using the extended syntax

$\texttt{not}\ \mathrm{K}\ p(X,Y) \ \texttt{or} \ \mathrm{M}\ q(X) \leftarrow d\_x(X), \ d\_y(Y), \ Y < 2.$

Grounding[11] the last rule results in four WV facts:

$\texttt{not}\ \mathrm{K}\ p(a,0) \ \texttt{or} \ \mathrm{M}\ q(a).$ $\texttt{not}\ \mathrm{K}\ p(b,0) \ \texttt{or} \ \mathrm{M}\ q(b).$

$\texttt{not}\ \mathrm{K}\ p(a,1) \ \texttt{or} \ \mathrm{M}\ q(a).$ $\texttt{not}\ \mathrm{K}\ p(b,1) \ \texttt{or} \ \mathrm{M}\ q(b).$

5 Examples and Simplifications

Henceforth, ES2016 extended with WVCs is assumed unless stated otherwise.

5.1 Epistemic Conformant Planning Module

The *epistemic conformant planning module*[12] for ES2014 with a sorted signature is as follows:

$occurs(A,S) \leftarrow \mathrm{M}\ occurs(A,S), \ S < n.$

$\neg occurs(A_2,S) \leftarrow occurs(A_1,S), \ A_1 \neq A_2.$

$success \leftarrow goal(n).$

$\leftarrow success, \ \texttt{not}\ \mathrm{K}\ success.$

$\leftarrow \texttt{not}\ \mathrm{M}\ success.$

where constant $n \in \mathbb{N}$ represents the plan horizon, variables A, A_1, and A_2 range over actions, and variable S ranges over integral time steps where $0 \leq S \leq n$. The last two rules are constraints that together (as discussed in Section 3) rule out world views that do not satisfy K *success*. With the proposed extension, we can replace these two constraints with one WVC that is succinct, intuitive, and easier to understand than the original pair of constraints:

---

[9] The associated *ground* domain atoms are understood to be the same in every belief set.

[10] It also fits well with the idea that the solver developer need not introduce a new token for the $\overset{\text{wv}}{\leftarrow}$ symbol.

[11] to include forward propagation with removal of body literals that are always *true* and removal of any rule where a body literal is always *false* (so-called "smart" grounding)

[12] See [22] for details on the use of ELPs to solve conformant planning problems using this module.

$\stackrel{\mathtt{wv}}{\leftarrow}$ $\mathtt{not}$ K $success$.
This is also relevant in that the proof of correctness for solving conformant planning problems encoded using the original epistemic conformant planning module (with the other elements of this methodology) depends in part on the two constraints ruling out world views that do not satisfy K *success*; however, that part of the proof is not valid for ES2016 semantics. Using the proposed WVC instead of the two original constraints elucidates this for both semantics.

5.2 Autonomous Control

Consider an exploratory robot operating on Mars with a round-trip communication delay of 30 minutes with Earth. Although an Earth operator may receive a continuous stream of data from the robot, the data is already 15 minutes old when received, and any instruction sent will not be received by the robot for another 15 minutes. As Thomas Ormston [32] of the European Space Agency put it, "there's a lot that can happen in half an hour on Mars." It is important, for example, that the robot does not fall off a cliff. Though intermittent goals may be provided from Earth, some autonomous control is needed for the robot to move at a reasonable pace. We envision as part of the on-board control system[13] of the robot an epistemic planning component that uses information about the terrain and observable surroundings to help form and select a plan to get to a specified goal. Included in rules used to plan could be WVCs as follows:

$\stackrel{\mathtt{wv}}{\leftarrow}$ M $likelihood\_of\_falling\_off\_a\_cliff(high)$.
$\stackrel{\mathtt{wv}}{\leftarrow}$ M $likelihood\_of\_falling\_off\_a\_cliff(moderate)$.
These would prevent selecting a plan where the possibility of falling off a cliff is high/moderate.

5.3 Subsumption and Simplification

In the table below are subjective literal forms that can subsume others in a rule body.[14]

subsumer	subsumed
K ℓ	M ℓ
	$\mathtt{not}$ M $\overline{\ell}$
	$\mathtt{not}$ K $\overline{\ell}$
M ℓ	$\mathtt{not}$ K $\overline{\ell}$
$\mathtt{not}$ M ℓ	$\mathtt{not}$ K ℓ

For example:
$\stackrel{\mathtt{wv}}{\leftarrow}$ K p, M p, $\mathtt{not}$ M $\neg p$, $\mathtt{not}$ K $\neg p$. $\equiv$ $\stackrel{\mathtt{wv}}{\leftarrow}$ K p.

$\stackrel{\mathtt{wv}}{\leftarrow}$ M p, $\mathtt{not}$ K $\neg p$. $\equiv$ $\stackrel{\mathtt{wv}}{\leftarrow}$ M p.
$\stackrel{\mathtt{wv}}{\leftarrow}$ $\mathtt{not}$ M p, $\mathtt{not}$ K p. $\equiv$ $\stackrel{\mathtt{wv}}{\leftarrow}$ $\mathtt{not}$ M p.

With world view constraints, subsumption can also occur across multiple rules, perhaps most easily seen using the WV fact form. Consider the following pair of WV facts:

K p. M p.

The subsumer-subsumed list above applies to pairs of non-disjunctive WV facts. Any world view satisfying the first rule must satisfy the second; thus, the second rule can be removed.

Identifying tautologies can also help in program simplification. For example, WV fact

K p $\mathtt{or}$ $\mathtt{not}$ K p.

is worthless and can be removed. For a more complex example, consider the following rules:

M q $\mathtt{or}$ K p. M q $\mathtt{or}$ $\mathtt{not}$ K p.

With respect to any world view of a program containing this pair, either Kp or $\mathtt{not}$ Kp will be satisfied (but not both), so these two rules can be reduced to the one rule: M q.

[13] Details of such a control system are beyond the scope of this paper and left to the reader's imagination.
[14] The symbol $\overline{\ell}$ in the table indicates the logical complement of (ground) objective literal ℓ; e.g., if $\ell = \neg p$ then $\overline{\ell} = p$. Logical subsumption follows from Definition 1 and the definition of a world view.

6 Algorithm for Computing World Views of an ELP with WVCs

The following is a generic algorithm for finding the world views of an ELP with WVCs:

Generic Algorithm

INPUT: Π (a ground ELP with WVCs)

1. partition Π into $\Pi_{\mathtt{wvc}}$ (the WVCs of Π) and $\Pi_0 = \Pi \setminus \Pi_{\mathtt{wvc}}$
2. use your favorite ELP solver to find the world views of Π_0
3. eliminate any world view of Π_0 that violates a WVC of $\Pi_{\mathtt{wvc}}$

OUTPUT: remaining world views of Π_0 not eliminated in Step 3

For those interested in implementing a solver, we now provide a more detailed algorithm. Details of an algorithm to compute the world views of an ELP under ES2016 semantics are given in [24]. We use a simplified version, modified to handle WVCs. Although we provided a grounding strategy for WVCs in Section 4.4, for brevity, the input is assumed ground.

Notation: From a ground ELP with WVCs $\Pi = \Pi_0 \cup \Pi_{\mathtt{wvc}}$, ASP program Π_0' is created as a modal reduct framework to aid in computing the world views of Π_0. For each literal ℓ appearing in an epistemic negation of the form `not` K ℓ in $E_P(\Pi_0)$, fresh atoms k_ℓ, $k0\_\ell$, $k1\_\ell$ are created by prefixing ℓ with $k\_$, $k0\_$, and $k1\_$ (respectively), and substituting 2 for $\neg$ if ℓ is a classically-/strongly-negated atom. Likewise, for ℓ appearing in an epistemic negation of the form M ℓ in $E_P(\Pi_0)$, fresh atoms m_ℓ, $m0\_\ell$, $m1\_\ell$ are created. These fresh atoms are referred to as *k-/m-atoms*, or, allowing for negated forms, *k-/m-literals*. For example, given an epistemic negation of the form `not` K ℓ, if $\ell = p(a)$ then k_ℓ denotes $k\_p(a)$, but if $\ell = \neg p(a)$ then k_ℓ denotes $k\_2p(a)$. Fresh atoms k_ℓ (in negated form) and m_ℓ are used as substitutes for K ℓ and M ℓ, respectively, in the ASP representation of the modal reduct of Π with respect to a potential world view. The intended meaning of $k1\_\ell$ is "K ℓ is *true*"; $k0\_\ell$ means "K ℓ is *false*"; $m1\_\ell$ means "M ℓ is *true*"; and $m0\_\ell$ means "M ℓ is *false*". Additionally, given a set W of sets of literals (including k-/m-literals), we use $W_{\setminus km}$ to denote *W modulo k-/m-literals* (i.e., the result of removing all k-/m-literals from sets in W).

The algorithm uses a "guess and check" method to compute the world views of Π_0. Each guess corresponds to a set of truth value assignments for the elements of $E_P(\Pi_0)$. A systematic approach is used, starting with the guess corresponding to the elements of $E_P(\Pi_0)$ being all *true*, working down by increasing the number of *false* elements by 1 at each successive level. Each computed world view of Π_0 is checked to ensure no WVC in $\Pi_{\mathtt{wvc}}$ is violated before it is considered a world view of Π. Any guess for which the epistemic negations assigned as *true* are a subset of those for a guess associated with a previously computed Π_0 world view will be filtered out. The order of computation and subsequent filtering enforces the maximality requirement of ES2016 semantics. (For ES2014, remove this filtering; also, computation order w.r.t. guesses is irrelevant.)

The algorithm iterates through all relevant guesses, one-guess-at-a-time, requiring (in general) computing answer sets of up to 2^n ASP programs where $n = |E_P(\Pi_0)|$. This is inefficient but relatively easy to understand. A more complex algorithm may involve including multiple guesses in each ASP program (at the expense of the need for aggregating computed answer sets) and parallelization. See [24] for a solver that uses this approach. Steps that handle WVCs can be applied there, as well as to other approaches, such as the one in [37].

Since we start with a proven algorithm for computing world views of Π_0, correctness of the algorithm is clear from the definitions and semantics of an ELP with WVCs given herein. We note that filtering out guesses that would violate WVCs *during* the computation of world views (rather than filtering out world views as a post-processing step as proposed here) could prune the search if $E_P(\Pi_0) \cap E_P(\Pi_{\mathtt{wvc}}) \neq \emptyset$, but would (in general) not be correct per ES2016 semantics. (That approach may be useful for an ES2014 solver.) If, however, there are epistemic negations in $\Pi_{\texttt{wvc}}$ that are not in Π_0, **the search space of guesses is pruned significantly from what it might be without WVCs**, assuming those from $\Pi_{\texttt{wvc}}$ would otherwise be included in other rules.

Finally, as the algorithm simply checks if world views of Π_0 violate the WVCs in $\Pi_{\texttt{wvc}}$, the effective complexity of solving Π is the same as for solving an *equivalent*[15] ELP Π_2 (without WVCs), assuming Π_2 differs only in constraints, with $|E_P(\Pi)| \leq |E_P(\Pi_2)|$.

Algorithm 1. [Computing the World Views of an ELP with WVCs]

INPUT: a ground ELP with WVCs Π	OUTPUT: the world views of Π

1. **Program Partition**: Partition Π into $\Pi_{\texttt{wvc}}$ (the WVCs of Π) and $\Pi_0 = \Pi \setminus \Pi_{\texttt{wvc}}$.
2. **Translation**: Create ASP program[16] Π'_0 from Π_0 by:
   - leaving rules without subjective literals unchanged;
   - otherwise, replacing subjective literals and adding new rules per the following table:

subj. lit. φ	**replace** φ **with**	**add rules**
$\mathrm{K}\,\ell$	`not` $\neg k\_\ell,\ \ell$	$\neg k\_\ell \leftarrow k0\_\ell.$
`not` $\mathrm{K}\,\ell$	$\neg k\_\ell$	$\neg k\_\ell \leftarrow k1\_\ell,$ `not` $\ell.$
$\mathrm{M}\,\ell$	m_ℓ	$m\_\ell \leftarrow m1\_\ell.$
`not` $\mathrm{M}\,\ell$	`not` m_ℓ	$m\_\ell \leftarrow m0\_\ell,$ `not not` $\ell.$

3. **Guess & Check:** Repeat (a)-(c) until all relevant guesses are generated and checked.
   a. **Generate Guess:** For each iteration, generate a guess Φ, starting with $\Phi = E_P(\Pi_0)$ for the first iteration and moving on in *popcount* order[17] for further iterations, filtering out any guess that is a subset of a guess associated with a previously found world view of Π_0. Create Π''_0 by appending to Π'_0 the ASP representation of Φ (i.e., k-/m-atoms as facts corresponding to the epistemic negations in Φ) as follows:
   $$\Pi''_0 = \Pi'_0 \cup \{k0\_\ell.\ \mid \texttt{not}\ \mathrm{K}\,\ell \in \Phi\} \cup \{k1\_\ell.\ \mid \texttt{not}\ \mathrm{K}\,\ell \in E_P(\Pi_0) \wedge \texttt{not}\ \mathrm{K}\,\ell \notin \Phi\}$$
   $$\cup\ \{m1\_\ell. \mid \mathrm{M}\,\ell \in \Phi\} \qquad \cup\ \{m0\_\ell. \mid \mathrm{M}\,\ell \in E_P(\Pi_0) \wedge \mathrm{M}\,\ell \notin \Phi\}.$$
   b. **Compute Answer Sets:** Use an ASP solver to compute the answer sets of Π''_0.
   c. **Check:** If Π''_0 is consistent, let W be the collection of answer sets computed in (b). Verify the following conditions:
      - if $k1\_\ell$ is in the sets of W, then ℓ is in every set of W;
      - if $k0\_\ell$ is in the sets of W, then ℓ is missing from at least one set of W;
      - if $m1\_\ell$ is in the sets of W, then ℓ is in at least one set of W; and
      - if $m0\_\ell$ is in the sets of W, then ℓ is missing from every set of W.

      $W_{\setminus km}$ is a world view of Π_0 if the conditions are met. $W_{\setminus km}$ **is a world view of** Π **if** $W_{\setminus km}$ **is a world view of** Π_0 **and** $W_{\setminus km}$ **doesn't violate any WVC in** $\Pi_{\texttt{wvc}}$.

7 Conclusions and Future Work

World view constraints provide a straightforward device for encoding restrictions on the world views of an ELP, allowing the specification of high-level conditions that must not be violated. They do not fix all semantics issues, but WVCs retain consistent meaning in all.

[15] We note there may not be a straightforward equivalent program without WVCs under ES2016 semantics.
[16] with nested expressions of the form `not not` ℓ as defined in [31]
[17] guess size ($|\Phi|$) will be reduced by one after exhausting all guesses of the current size

WVCs can also be a useful addition to languages extending Epistemic Specifications, such as ASP$^{\mathsf{GM}}$ [46]. Subjective literals in ASP$^{\mathsf{GM}}$ have the form $\mathrm{M}_{[lb:ub]}\,\ell$ where $lb, ub \in \mathbb{N}$, $lb \leq ub$, and ℓ is a literal. For $\mathrm{M}_{[lb:ub]}\,\ell$ to be satisfied by a world view, the number of belief sets containing ℓ must be in the closed range $[lb, ub]$. To indicate no upper bound, ub can be omitted. The definitions in Section 4.3.1 can be used to extend ASP$^{\mathsf{GM}}$ with WVCs; however, support for negated subjective literals needs to be added for the later definitions to apply.

For future work, we would like to incorporate the notion of *weak* WVCs into Epistemic Specifications in a manner analogous to *weak constraints* [10] in ASP, but at the world view level. In principle, these would function like normal WVCs unless the program is inconsistent, where they would be systematically relaxed (perhaps in order by given weight/level) until consistency or exhaustion. Ergo, we introduce symbol $\overset{\mathsf{wv}}{\leftsquigarrow}$ and suggest the following syntax:

$\overset{\mathsf{wv}}{\leftsquigarrow} s_1,\ \ldots,\ s_n.\ [w@l]$

where $n > 0$, each s_i is a (possibly negated) subjective literal, and both w and l are non-negative integers representing *weight* and *level* values, respectively. Returning to the Martian robot example of Section 5.2, it may be more appropriate for planning to use the following:

$\xleftarrow{\mathsf{wv}}$ M $likelihood\_of\_falling\_off\_a\_cliff(high)$.
$\overset{\mathsf{wv}}{\leftsquigarrow}$ M $likelihood\_of\_falling\_off\_a\_cliff(moderate)$. [1@0]

These rules express a preference for plans where likelihood of falling off a cliff is neither high nor moderate, but if none exist, moderate likelihood is accepted by relaxing the weak WVC.

References

1. Evgenii Balai. `ELPS`, 2015. Texas Tech. URL: `https://github.com/iensen/elps/wiki/`.
2. Evgenii Balai and Patrick Kahl. Epistemic logic programs with sorts. In Daniela Inclezan and Marco Maratea, editors, *Proc. 7th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2014)*. URL: `https://sites.google.com/site/aspocp2014/paper_4.pdf`.
3. Marcello Balduccini. `sismodels`, 2001. See `http://www.mbal.tk/` for more information.
4. Marcello Balduccini and Tran Cao Son, editors. *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, volume 6565 of *LNCS*. Springer, 2011. `doi:10.1007/978-3-642-20832-4`.
5. Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, New York, NY, USA, 2003.
6. Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *J. Log. Program.*, 19/20:73–148, 1994. `doi:10.1016/0743-1066(94)90025-6`.
7. Manuel Bichler, Michael Morak, and Stefan Woltran. `selp`, 2018. URL: `http://dbai.tuwien.ac.at/proj/selp/`.
8. Manuel Bichler, Michael Morak, and Stefan Woltran. Single-shot epistemic logic program solving. In Jérôme Lang, editor, *Proc. 27th Intl. Joint Conf. on AI (IJCAI 2018)* [to appear], 2018. URL: `http://dbai.tuwien.ac.at/proj/selp/ijcai2018.pdf`.
9. Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczyński. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011. `doi:10.1145/2043174.2043195`.
10. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Strong and weak constraints in disjunctive datalog. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proc. 4th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'97)*, volume 1265 of *LNCS*, pages 2–17. Springer, 1997. `doi:10.1007/3-540-63255-7_2`.
11. Jianhua Chen. The generalized logic of only knowing (GOL) that covers the notion of epistemic specifications. *J. Log. Comput.*, 7(2):159–174, 1997. `doi:10.1093/logcom/7.2.159`.

12 Rongcun Cui, Zhizheng Zhang, and Kaikai Zhao. ESParser: An epistemic specification grounder. In James P. Delgrande and Wolfgang Faber, editors, *Proc. 1st Intl. Conf. on Computer Science and Service System (CSSS 2012)*, pages 1823–1827. IEEE Computer Society CPS, 2012. doi:10.1109/CSSS.2012.454.

13 Luis Fariñas del Cerro, Andreas Herzig, and Ezgi Iraz Su. Epistemic equilibrium logic. In Qiang Yang and Michael Wooldridge, editors, *Proc. 24th Intl. Joint Conf. on AI (IJCAI 2015)*, pages 2964–2970. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/419.

14 Wolfgang Faber and Stefan Woltran. Manifold answer-set programs and their applications. In Balduccini and Son [4], pages 44–63. doi:10.1007/978-3-642-20832-4_4.

15 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice.* Synthesis Lectures on AI and ML. Morgan and Claypool, 2012.

16 Michael Gelfond. Strong introspection. In Thomas L. Dean and Kathleen McKeown, editors, *Proc. 9th National Conf. on Artificial Intelligence (AAAI-91)*, pages 386–391. AAAI/MIT Press, 1991. URL: http://www.aaai.org/Papers/AAAI/1991/AAAI91-060.pdf.

17 Michael Gelfond. Logic programming and reasoning with incomplete information. *Ann. Math. Artif. Intell.*, 12(1-2):89–116, 1994. doi:10.1007/BF01530762.

18 Michael Gelfond. New semantics for epistemic specifications. In James P. Delgrande and Wolfgang Faber, editors, *Proc. 11th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2011)*, volume 6645 of *LNCS*, pages 260–265. Springer, 2011. doi:10.1007/978-3-642-20895-9_29.

19 Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach.* Cambridge University Press, 2014. doi:10.1017/CBO9781139342124.

20 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proc. 5th Intl. Conf. and Symposium on Logic Programming (ICLP/SLP 1988)*, pages 1070–1080. MIT Press, 1988.

21 Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991. doi:10.1007/BF03037169.

22 Patrick Kahl, Richard Watson, Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation*, 2015. doi:10.1093/logcom/exv065.

23 Patrick Thor Kahl. *Refining the Semantics for Epistemic Logic Programs.* PhD thesis, Texas Tech, Lubbock, TX, USA, May 2014. URL: http://hdl.handle.net/2346/58710.

24 Patrick Thor Kahl, Anthony P. Leclerc, and Tran Cao Son. A parallel memory-efficient epistemic logic program solver: Harder, better, faster. In Bart Bogaerts and Amelia Harrison, editors, *Proc. 9th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2016)*, 2016. URL: https://arxiv.org/abs/1608.06910.

25 Michael Kelly. *Wviews: A World View Solver for Epistemic Logic Programs*, October 2007.

26 Michael Kelly. Wviews, 2018. URL: https://github.com/galactose/wviews.

27 Tiep Le and Tran Cao Son. EP-ASP, 2017. NMSU. URL: https://github.com/tiep/EP-ASP.

28 Anthony P. Leclerc and Patrick Thor Kahl. A survey of advances in epistemic logic program solvers. In Jorge Fandinno and Johannes K. Fichte, editors, *Proc. 11th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2018)* [to appear], 2018.

29 Hector J. Levesque. All I know: A study in autoepistemic logic. *Artif. Intell.*, 42(2-3):263–309, 1990. doi:10.1016/0004-3702(90)90056-6.

30 Vladimir Lifschitz. What is answer set programming? In Dieter Fox and Carla P. Gomes, editors, *Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008)*, pages 1594–1597.

31 Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Ann. Math. Artif. Intell.*, 25(3-4):369–389, 1999. doi:10.1023/A:1018978005636.

32 Thomas Ormston. Time delay between Mars and Earth. In: ESA's *Mars Express* blog. URL: `http://blogs.esa.int/mex/2012/08/05/time-delay-between-mars-and-earth/`.

33 David Pearce. A new logical characterisation of stable models and answer sets. In Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors, *Proc. Non-Monotonic Extensions of Logic Programming (NMELP 1996)*, volume 1216 of *LNCS*, pages 57–70. Springer, 1996. `doi:10.1007/BFb0023801`.

34 Mircea Preda. Modeling epistemic knowledge in logic programs with negation as failure. In Dimitris Dranidis and Illias Sakellariou, editors, *Proc. 3rd South-East European Workshop on Formal Methods (SEEFM'07)*. SEERC, 2007.

35 SEU. `ESmodels`, 2015. URL: `http://cse.seu.edu.cn/people/seu_zzz/indexe.htm`.

36 Yi-Dong Shen and Thomas Eiter. Evaluating epistemic negation in answer set programming. *Artif. Intell.*, 237:115–135, 2016. `doi:10.1016/j.artint.2016.04.004`.

37 Tran Cao Son, Tiep Le, Patrick Kahl, and Anthony Leclerc. On computing world views of epistemic logic programs. In Carles Sierra, editor, *Proc. 26th Intl. Joint Conf. on AI (IJCAI 2017)*, pages 1269–1275. `doi:10.24963/ijcai.2017/176`.

38 Anton Strasser. `EHEX`, 2018. TU Wien. URL: `https://github.com/hexhex/ehex`.

39 Tommi Syrjänen and Ilkka Niemelä. The Smodels system. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczyński, editors, *Proc. 6th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2001)*, volume 2173 of *LNCS*, pages 434–438. Springer, 2001. `doi:10.1007/3-540-45402-0_38`.

40 Miroslaw Truszczyński. Revisiting epistemic specifications. In Balduccini and Son [4], pages 315–333. `doi:10.1007/978-3-642-20832-4_20`.

41 UWS. `Wviews`, 2007. URL: `http://staff.scem.uws.edu.au/~yan/Wviews.html`.

42 Kewen Wang and Yan Zhang. Nested epistemic logic programs. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *Proc. 8th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, volume 3662 of *LNCS*, pages 279–290. Springer, 2005. `doi:10.1007/11546207_22`.

43 Richard Glenn Watson. *An Inference Engine for Epistemic Specifications*, May 1994.

44 Yan Zhang. Computational properties of epistemic logic programs. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proc. 10th Intl. Conf. on Principles of Knowledge Representation and Reasoning*, pages 308–317. AAAI Press, 2006.

45 Yan Zhang and Yuanlin Zhang. Epistemic specifications and conformant planning. In Roman Barták, Thomas Leo McCluskey, and Enrico Pontelli, editors, *Proc. 2017 Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS 2017).*

46 Zhizheng Zhang. Answer set programming with graded modality. In Marcello Balduccini and Tomi Janhunen, editors, *Proc. 14th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2017)*, volume 10377 of *LNCS*, pages 205–211. Springer, 2017. `doi:10.1007/978-3-319-61660-5_18`.

47 Zhizheng Zhang, Kaikai Zhao, and Rongcun Cui. ESmodels: An inference engine of epistemic specifications. In *Proc. 25th Intl. Conf. on Tools with Artificial Intelligence (ICTAI 2013)*, pages 769–774. IEEE Computer Society, 2013. `doi:10.1109/ICTAI.2013.118`.