# On Computing World Views of Epistemic Logic Programs

**Tran Cao Son** and **Tiep Le**
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA

**Patrick Kahl** and **Anthony Leclerc**
SPAWAR Atlantic
North Charleston, SC 29410, USA

## Abstract

This paper presents a novel algorithm for computing world views of different semantics of epistemic logic programs (ELP) and two of its realization, called EP-ASP (for an older semantics) and EP-ASP$^{se}$ (for the newest semantics), whose implementation builds on the theoretical advancement in the study of ELPs and takes advantage of the multi-shot computation paradigm of the answer set solver CLINGO. The new algorithm differs from the majority of earlier algorithms in its strategy. Specifically, it computes one world view at a time and utilizes properties of world views to reduce its search space. It starts by computing an answer set and then determines whether or not a world view containing this answer set exists. In addition, it allows for the computation to focus on world views satisfying certain properties. The paper includes an experimental analysis of the performance of the two solvers comparing against a recently developed solver. It also contains an analysis of their performance in goal directed computing against a logic programming based conformant planning system, DLV-K. It concludes with some final remarks and discussion on the future work.

## 1 Introduction

Epistemic logic programs (ELP) were proposed [Gelfond, 1991; 1994] for introspective reasoning through the use of modal operators K ("known") and M ("may be true"). For example, consider the following rules that were told to security guards of an extremely secret office:

- If you know that an individual breaks the data transmission rule then put him in a maximal secure prison.
- If you suspect that an individual breaks the data transmission rule but does not know that he/she breaks the rule then take him into custody and interview him.

If we were to encode these rules as a logic program that would be used by a guard to determine whether or not an individual, given a set of observations, is a felon or a suspect, which will then allow the guard to act appropriately. Let *felon*, *suspect*, and *break_rule* denote that an individual is a felon, is a suspect, and breaks the rule, respectively. Then, an encoding of these rules in ELP is as follows:

$$felon \leftarrow \mathrm{K}\, break\_rule \tag{1}$$

$$suspect \leftarrow \mathrm{M}\, break\_rule,\ \mathtt{not}\ \mathrm{K}\, break\_rule \tag{2}$$

where the K (resp. M) operator stands for *know* (resp. *might know*) respectively. The intuitive meaning of K is that if $break\_rule$ is true in every belief set of the agent then K $break\_rule$ is true and hence $felon$ will be concluded. And, if $break\_rule$ is true in some but not all belief sets of the agent then M $break\_rule$ is true and hence $suspect$ will be concluded. Observe that this reasoning process requires that the truth value of epistemic literals such as K $break\_rule$ and M $break\_rule$ is evaluated over *all belief sets* of the agent.

In the past two decades, several semantics for ELPs have been proposed [Gelfond, 1991; 1994; del Cerro *et al.*, 2015; Kahl *et al.*, 2015; Truszczyński, 2011; Shen and Eiter, 2016]. Roughly speaking, the semantics of an ELP program $\Pi$ is defined by the notion of a *world view* which is a collection of answer sets of a logic program $\Pi'$ without the literals containing the K and M operators (or subjective literals) obtained after a modal reduct of $\Pi$ with respect to a set of literals (precise definition in the next section). As such, with the exception of the algorithm in [Kahl *et al.*, 2015], other algorithms for computing world views follow a guess-and-check style algorithm and employ two steps: (*i*) guess the values of a set of literals for the modal reduct; (*ii*) compute the modal reduct, its answer sets using some answer set solver, and verify that the values of guessed literals with respect to the collection of answer sets of the modal reduct match the guessed values. The difference between these algorithms lies in which type of literals are selected for guessing and how their values are guessed. For instance, the algorithm developed in [Zhang and Zhao, 2014] starts with guessing the values of objective literals occurring in the subjective literals of the programs; the algorithm proposed in [Zhang, 2006] starts with the subjective literals; etc. The algorithm developed in [Kahl *et al.*, 2015] starts with transforming an ELP program into a logic program without guessing the values of the subjective literals. It then searches for world views among the collection of answer sets of this program.

It is worth noticing that thus far, with the exception of the recently developed system `ELPsolve` described in [Kahl *et al.*, 2016] that can compute world views of programs with millions of possible assignments of the subjective literals, none of the previously developed systems reaches this level of scalability. Perhaps the earliest attempt to develop an ELP-solver is the system `Wviews` described in [Kelly, 2007]. This system implemented the algorithm proposed in [Zhang, 2006]. `ESmodels` (see, e.g., [Zhang and Zhao, 2014]) is another system for computing world views that implements a guess-and-check algorithm that starts with guessing of values of literals occurring in subjective literals. The system also utilized some properties of world views to enhance its performance. `ELPS` [Balai and Kahl, 2014] is yet another system for computing an improved semantics of ELPs, called ES2014, proposed in [Kahl *et al.*, 2015]. The difference between `ELPS` and `ELPsolve` lies in that `ELPS` computes *all* answer sets of the program resulting from the transformation mentioned earlier while `ELPsolve` implements a systematic search over the set of truth value assignments of the set of subjective literals that occur within the scope of a negation-as-failure operator. In this way, `ELPsolve` avoids the memory problem that frequently plagued other systems. `ELPsolve` also exploits parallelism to improve its performance. We observe that all previous developed ELP-solvers usually employ answer set solvers as a black box.

Our goal in this paper is to contribute to the development of ELP-solvers by proposing new algorithms that facilitate a tighter integration of world view computation and answer set solvers. We demonstrate the usefulness of the algorithms by developing two novel ELP-solvers for computing world views and show experimentally that the new solvers perform better than the state-of-the-art solvers. We also evaluate the use of these solvers in conformant planning by comparing it against DLV-K, one of the most efficient logic programming based conformant planners.

## 2 Background

We assume that the readers are familiar with the basic notations of logic programs under answer set semantics (or *ASP programs*) (see, e.g., [Gelfond and Lifschitz, 1991]). For an ASP program $\mathcal{P}$, AS($\mathcal{P}$) denotes the set of all answer sets of $\mathcal{P}$. Answer sets of ASP programs can be computed using ASP solvers such as CLINGO [Gebser *et al.*, 2014]. Recently, CLINGO adds a feature, called multi-shot, for dealing with continuously changing logic programs. It is done by augmenting an ASP encoding with Python procedures controlling ASP solving processes along with the corresponding evolving logic programs. It includes methods for adding/grounding rules, setting truth values of (external) atoms, computing the answer sets of a current program, etc.

We assume a propositional language $\mathcal{L}$. An *objective literal* (or *literal*, for short) $\ell$ in $\mathcal{L}$ is either an atom $a \in \mathcal{L}$ or its negation $\neg a$. A *subjective literal* in $\mathcal{L}$ is of the form K$\ell$, M$\ell$, $\texttt{not}$ K$\ell$, or $\texttt{not}$ M$\ell$ where $\ell$ is an objective literal. An ELP is a set of rules of the form

$$\ell_1 \texttt{ or } ... \texttt{ or } \ell_k \leftarrow g_1, ..., g_m, \texttt{ not } g_{m+1}, ..., \texttt{ not } g_n.$$

where $k \geq 0$, $m \geq 0$, $n \geq m$, each $\ell_i$ is a literal, and each $g_i$ is either a literal, or a subjective literal. Observe that an ELP that does not contain any occurrence of a subjective literal is an ASP program. We say that a set of literals in $\mathcal{L}$ is *consistent* if it does not contain $a$ and $\neg a$ for some $a \in \mathcal{L}$.

**Definition 1 (Satisfaction of a Subjective Literal)** Let $W$ be a non-empty set of consistent sets of literals, and $\varphi$ be a subjective literal. $W$ *satisfies* $\varphi$, denoted by $W \models \varphi$ if

- $\varphi = \mathrm{K}\ell$ and $\forall A \in W : \ \ell \in A$.
- $\varphi = \texttt{not }\mathrm{K}\ell$ and $\exists A \in W : \ \ell \notin A$.
- $\varphi = \mathrm{M}\ell$ and $\exists A \in W : \ \ell \in A$.
- $\varphi = \texttt{not }\mathrm{M}\ell$ and $\forall A \in W : \ \ell \notin A$.

**Definition 2 (Modal Reduct)** Let $\Pi$ be an ELP and $W$ be a non-empty set of consistent sets of ground literals. $\Pi^W$, the *modal reduct of* $\Pi$ *with respect to* $W$, is defined as the ASP program[1] obtained from $\Pi$ by replacing/removing subjective literals and/or deleting associated rules in $\Pi$ as follows:

- $\varphi = \mathrm{K}\ell$ and $W \models \varphi$ then replace K$\ell$ with $\ell$; otherwise, delete the rule from $\Pi$;
- $\varphi = \texttt{not }\mathrm{K}\ell$ and $W \models \varphi$ then remove $\texttt{not}$ K$\ell$ from the rule; otherwise, replace $\texttt{not}$ K$\ell$ with $\texttt{not}$ $\ell$;
- $\varphi = \mathrm{M}\ell$ and $W \models \varphi$ then remove M$\ell$ from the rule; otherwise, replace M$\ell$ with $\texttt{not not}$ $\ell$;
- $\varphi = \texttt{not }\mathrm{M}\ell$ and $W \models \varphi$ then replace $\texttt{not}$ M$\ell$ with $\texttt{not}$ $\ell$; otherwise, delete the rule from $\Pi$.

**Definition 3 (World View)** Let $\Pi$ be a ground ELP and $W$ be a non-empty set of consistent sets of literals. $W$ is a *world view* of $\Pi$ if $W = \mathrm{AS}(\Pi^W)$.

An ELP program might have multiple world views. For example, the program

$$p \texttt{ or } q. \quad r \leftarrow \texttt{not } \mathrm{M}p. \quad \neg p \leftarrow \mathrm{M}r, \texttt{ not } q.$$

has two world views $\{\{p\}, \{q\}\}$ and $\{\{q, r\}\}$.

Given an ELP $\Pi$, world views of $\Pi$ can be computed using an algorithm in [Kahl *et al.*, 2015]. For simplicity of the presentation, the algorithm is simplified to only deal with ground programs.

- **Step 1**: Convert $\Pi$ to a program $\Pi'$ without K and M operators.
  For each objective literal $\ell$ occurring in a subjective literal, let $k\_\ell$, $k0\_\ell$, $k1\_\ell$, $m\_\ell$, $m0\_\ell$, or $m1\_\ell$ be fresh atoms[2] obtained from $\ell$ by prefixing it with $k\_$, $k0\_$, $k1\_$, $m\_$, $m0\_$, or $m1\_$ (respectively). These atoms are referred to as *k-/m-atoms*. Two sets $S_1$ and $S_2$ of literals with literals from these fresh atoms are said to share the same k-/m-atoms, denoted by $S_1 \sim_{k/m} S_2$ if $x\_\ell \in S_1$ iff $x\_\ell \in S_2$ for all $x \in \{k, k0, k1, m, m0, m1\}$ and objective literal $\ell$.
  Let $\Pi'$ be the program that consists of rules without subjective literals in $\Pi$ and for each rule $r$ in $\Pi$ that contains some subjective literals, replace the subjective literals in $r$ to create a new rule $r'$ in $\Pi'$ as follows[3].
  - K$\ell$ is replaced with $\texttt{not } \neg k\_\ell, \ell$ and the following rules will be added to $\Pi'$:

[1]with nested expressions of the form $\texttt{not not}$ $\ell$ (see, [Lifschitz *et al.*, 1999])

[2]Atoms that do not occur in $\mathcal{L}$.

[3]Note that if a rule $r$ contains more than one subjective literals then multiple replacements/deletions and multiple sets of rules are introduced in the creation of $r'$.

$k1\_\ell \leftarrow \texttt{not}\ k0\_\ell.$ (3) $\quad \neg k\_\ell \leftarrow k0\_\ell.$ (5)

$k0\_\ell \leftarrow \texttt{not}\ k1\_\ell.$ (4) $\quad \neg k\_\ell \leftarrow k1\_\ell,\ \texttt{not}\ \ell.$ (6)

- ∘ $\texttt{not}$ K$\ell$ is replaced with $\neg k\_\ell$ and the rules (3)–(6) are added to $\Pi'$.
- ∘ M$\ell$ is replaced with $m\_\ell$ and the following rules will be added to $\Pi'$:

$m1\_\ell \leftarrow \texttt{not}\ m0\_\ell.$ (7) $\quad m\_\ell \leftarrow m1\_\ell.$ (9)

$m0\_\ell \leftarrow \texttt{not}\ m1\_\ell.$ (8) $m\_\ell \leftarrow m0\_\ell,\ \texttt{not not}\ \ell.$ (10)

- ∘ $\texttt{not}$ M$\ell$ is replaced with $\texttt{not}\ m\_\ell$ and the rules (7)-(9) and the next rule will be added to $\Pi'$:

$$m\_\ell \leftarrow m0\_\ell, \ell. \quad (11)$$

- **Step 2**: Group the answer sets of $\Pi'$ by common k-/m-atoms of the form $k0\_\ell$, $k1\_\ell$, $m0\_\ell$, and $m1\_\ell$, i.e., for each group of answer sets $W$, it holds that $\forall\ S_1,\ S_2 \in W.[S_1 \sim_{k/m} S_2]$. Each group $W$ of answer sets is said to *represent* (i.e., if its k-/m-literals are removed) a *candidate world view* of $\Pi$.
- **Step 3**: For each $W$ representing a candidate world view of $\Pi$, check that the following conditions are met for all its k-/m-atoms:
  (a) if $k1\_\ell$ is in the sets of $W$, then $\ell$ is in every set of $W$;
  (b) if $k0\_\ell$ is in the sets of $W$, then $\ell$ is missing from at least one set of $W$;
  (c) if $m1\_\ell$ is in the sets of $W$, then $\ell$ is in at least one set of $W$; and
  (d) if $m0\_\ell$ is in the sets of $W$, then $\ell$ is missing from every set of $W$.

  $W_{\setminus\text{KM}} = \{S_{\setminus\text{KM}} \mid S \in W\}$ is a *world view* of $\Pi$ if the conditions (a)–(d) above are met where $S_{\setminus\text{KM}}$ is obtained from $S$ by removing all fresh atoms from $S$.

In the following, we will denote with $ASP(\Pi)$ the program obtained from $\Pi$ after **Step 1** of the above algorithm.

## 3 Algorithms for Computing World Views

In this section, we describe three algorithms. One algorithm computes at most $k$ world views of an ELP. The second/third one computes a world view that satisfies a certain objective literal without/with using heuristics, respectively. We begin with a discussion of the theoretical foundation underlying these algorithms which depends on several properties of $ASP(\Pi)$. Before we present these properties, let us recall two important entailment notions in ASP. Given an ASP program $\mathcal{P}$ and a literal $\ell$ in the language of $\mathcal{P}$, $\ell$ is said to be *skeptically* (resp. *credulously*) entailed by $\mathcal{P}$, denoted by $\mathcal{P} \models_S \ell$ (resp. $\mathcal{P} \models_C \ell$), if $\ell \in S$ for every $S \in \text{AS}(\mathcal{P})$ (resp. for some $S \in \text{AS}(\mathcal{P})$).

Because a world view of an epistemic program $\Pi$ is a non-empty subset of AS$(ASP(\Pi))$, the next property is trivial.

**Proposition 1** *Let $\Pi$ be an epistemic program. If $ASP(\Pi)$ has no answer set then $\Pi$ has no world view.*

The next property makes use of the properties of the skeptical and credulous entailment relation in ASP programs.

**Proposition 2** *Let $\Pi$ be an epistemic program. Then,*

- *For an objective literal $\ell$ such that $ASP(\Pi)$ skeptically entails $\ell$ then K$\ell$ must be true in all world views of $\Pi$.*
- *For an objective literal $\ell$ such that $ASP(\Pi)$ does not credulously entail $\ell$ then* $\texttt{not}$ *M$\ell$ must be true in all world views of $\Pi$.*

The above two propositions are useful in preprocessing $\Pi$ since ASP solver provides options for computing skeptically and credulously entailment consequences of an ASP program (e.g., `cautious` and `brave` options of CLINGO). For example, if $\ell$ is skeptically entailed by $ASP(\Pi)(\Pi)$ then $k1\_\ell$ must be in every world view of $\Pi$. In order to present other properties, we need some extra notations. Let $X$ be a set of literals in the language of $ASP(\Pi)$. We define

$k1\_m0(X) = \{k1\_\ell \mid k1\_\ell \in X\} \cup \{m0\_\ell \mid m0\_\ell \in X\}$ and
$k0\_m1(X) = \{k0\_\ell \mid k0\_\ell \in X\} \cup \{m1\_\ell \mid m1\_\ell \in X\}$.

To continue, we introduce some new rules involving the atoms of the form $k1\_\ell$, $k0\_\ell$, $m1\_\ell$, or $m0\_\ell$ as follows. In the following, $nok$ is a fresh atom as well as $ok_\alpha$, for each k-/m- atom $\alpha$, is a fresh atom in the language of $ASP(\Pi)$.

| $\alpha$ is | Name | Rule/Constraint |
|---|---|---|
| $k1\_\ell$, $k0\_\ell$, $m1\_\ell$ , $m0\_\ell$ | $r_\alpha$ | $\leftarrow \texttt{not}\ \alpha.$ |
| $k1\_\ell$ | $c_\alpha$ | $nok \leftarrow \alpha,\ \texttt{not}\ \ell.$ |
| $m0\_\ell$ | $c_\alpha$ | $nok \leftarrow \alpha, \ell.$ |
| $k0\_\ell$ | $c_\alpha$ | $ok_\alpha \leftarrow \alpha,\ \texttt{not}\ \ell.$ |
| $m1\_\ell$ | $c_\alpha$ | $ok_\alpha \leftarrow \alpha, \ell.$ |

The next observation will be important in the proofs of the useful properties of $ASP(\Pi)$. Let $km(\Pi)$ be the set of all expressions of the form K$\ell$ or M$\ell$ occurring in $\Pi$.

**Observation 1** *Let $\Pi$ be an epistemic logic program and $S$ be an answer set of $ASP(\Pi)$. Then, for every K$\ell$ (resp. M$\ell$) that appears in $\Pi$, $S$ contains exactly one element of $\{k1\_\ell, k0\_\ell\}$ (resp. $\{m1\_\ell, m0\_\ell\}$).*

The correctness of this observation rests on the rules (3)-(4) and (7)-(8) in $ASP(\Pi)$ which are responsible for generating the fresh atoms in answer sets of $ASP(\Pi)$. Furthermore, a consequence of this observation and the definition of world views of $\Pi$ constructed from $ASP(\Pi)$, if a world view $W$ containing an answer set $S$ of $ASP(\Pi)$ exists then every answer set $S' \in W$ must satisfy that $k1\_m0(S) = k1\_m0(S')$ and $k0\_m1(S) = k0\_m1(S')$. The next proposition relates world views of $\Pi$ and answer sets of $ASP(\Pi)$ through the atoms in $km(\Pi)$.

**Proposition 3** *Let $\Pi$ be an epistemic program and $S$ be an answer set of $ASP(\Pi)$. Let $\Delta = \bigcup_{\alpha \in k1\_m0(S)} \{r_\alpha, c_\alpha\} \cup \bigcup_{\alpha \in k0\_m1(S)} \{r_\alpha\} \cup \{\leftarrow \texttt{not}\ nok\}$. Let $\Pi' = ASP(\Pi) \cup \Delta$. Then, if $\Pi'$ has some answer set then $\Pi$ does not have a world view containing $S_{\setminus\text{KM}}$.*

The above proposition allows for the development of a test that checks whether or not $\Pi$ has a world view containing an answer set $S$ of $ASP(\Pi)$. This is encoded in Algorithm 1. Specifically, given $S$, the set $\Delta$, defined in Proposition 3, is computed (Line 4) and an answer set of $ASP(\Pi) \cup \Delta$ is computed (Line 5). The rules in $\Delta$ stipulate that answer sets of $ASP(\Pi) \cup \Delta$ belong to the same group $W$ of answer sets

**Algorithm 1** TESTK1M0$(ASP(\Pi), S)$
*% Can* $\Pi$ *have a world view containing* $S$*?*

```
1: Input: $ASP(\Pi)$: ASP program of $\Pi$, $S \in AS(ASP(\Pi))$
2: Output: $\Pi$ has world view $W$, $S \in W$: true; else false
3: Compute $k1\_m0(S)$ and $k0\_m1(S)$
4: $\Delta = \bigcup_{\alpha \in k1\_m0(S)} \{r_\alpha, c_\alpha\} \cup \bigcup_{\alpha \in k0\_m1(S)} \{r_\alpha\} \cup \{\leftarrow \texttt{not } nok\}$.
5: if $ASP(\Pi) \cup \Delta$ has an answer set then return false
6: else return true
```

containing $S$ (**Step 2**, Section 2) and contain $nok$. The presence of $nok$ indicates that there exists some k-/m-atom $\alpha$ in $S$ such that the check for condition (a) and (c) in **Step 3** (Section 2) is not satisfied (e.g., $k1\_\ell$ is in the sets of $W$ but $\ell$ is not).

The next proposition is similar to Proposition 3. It provides a check for conditions (b) and (d) in verifying whether a candidate world view is indeed a world view.

**Proposition 4** *Let* $\Pi$ *be an epistemic program and* $S$ *be an answer set of* $ASP(\Pi)$*. Let* $\Delta = \bigcup_{\alpha \in k1\_m0(S)} \{r_\alpha\} \cup \bigcup_{\alpha \in k0\_m1(S)} \{r_\alpha, c_\alpha\}$*. Let* $\Pi' = ASP(\Pi) \cup \Delta$*. Then, if* $\Pi'$ *does not credulously entail* $ok_\alpha$ *for some* $\alpha \in k0\_m1(S)$ *then* $\Pi$ *does not have a world view containing* $S_{\setminus \text{KM}}$.

Algorithm 2 encodes Proposition 4. Similar to Algorithm 1, given $S$, the set $\Delta$, defined in Proposition 4, is computed (Line 4). Line 5 of the algorithm aims at enforcing conditions (b) and (d) (**Step 3**, Section 2) for the collection of answer sets of $ASP(\Pi) \cup \Delta$.

**Algorithm 2** TESTK0M1$(ASP(\Pi), S)$
*% Can* $\Pi$ *have a world view containing* $S$*?*

```
1: Input: $ASP(\Pi)$: ASP program of $\Pi$, $S \in AS(ASP(\Pi))$
2: Output: $\Pi$ has world view $W$, $S \in W$: true; else false
3: Compute $k1\_m0(S)$ and $k0\_m1(S)$
4: $\Delta = \bigcup_{\alpha \in k1\_m0(S)} \{r_\alpha\} \cup \bigcup_{\alpha \in k0\_m1(S)} \{r_\alpha, c_\alpha\}$.
5: if $ASP(\Pi) \cup \Delta \not\models_C ok_\alpha$ for some $\alpha \in k0\_m1(S)$ then
   return false
6: else return true
```

Utilizing Propositions 2–4, we can prove the following property of views of ELPs (proof omitted for space reason):

**Proposition 5** *Let* $\Pi$ *be an epistemic program,* $S$ *be an answer set of* $ASP(\Pi)$*, and* $\Delta = \{r_\alpha \mid \alpha \in k1\_m0(S) \cup k0\_m1(S)\}$*. If* TESTK1M0$(ASP(\Pi), S)$ *returns* `true` *and* TESTK0M1$(ASP(\Pi), S)$ *returns* `true` *then* $W_{\setminus \text{KM}}$ *is a world view of* $\Pi$ *where* $W = \{S \mid S \in AS(ASP(\Pi) \cup \Delta)\}$.

## 3.1 Computing $k$ World Views

In this subsection, we describe an algorithm for computing at most $k$ world views of an epistemic logic program. We need also the following notations. For a program $\Pi$ and $S \in AS(ASP(\Pi))$, by $C(S)$ we denote the constraint

$$\leftarrow \alpha_1, \ldots, \alpha_t \tag{12}$$

where $\{\alpha_1, \ldots, \alpha_t\} = k1\_m0(S) \cup k0\_m1(S)$. Also,

$$S_\Pi = \{\leftarrow \texttt{not } k1\_\ell \mid ASP(\Pi) \models_S \ell\} \tag{13}$$

**Algorithm 3** EP-ASP$(\Pi, k)$: Compute $k$ world views of $\Pi$

```
1: Input: an ELP program$\Pi$, $k$: the number of world views
2: Output: At most $k$ world views of $\Pi$
3: Compute $ASP(\Pi)$ from $\Pi$
4: Let $ASP(\Pi) = ASP(\Pi) \cup S_\Pi \cup C_\Pi$
5: $\Omega = \emptyset$; $\Gamma = ASP(\Pi)$; $n = 0$
6: while $n < k$ and $\Gamma$ is consistent do
7:    Let $S$ be an answer set of $\Gamma$
8:    $\Delta = \{r_\alpha \mid \alpha \in k1\_m0(S) \cup k0\_m1(S)\}$
9:    if TESTK1M0$(ASP(\Pi), S)$ = true then
10:      if TESTK0M1$(ASP(\Pi), S)$ = true then
11:         $W = \mathcal{AS}(ASP(\Pi) \cup \Delta)$
12:         $\Omega = \Omega \cup W_{\setminus \text{KM}}$; $n = n + 1$
13:   $\Gamma = \Gamma \cup C(S)$
14: end while
15: return $\Omega$
```

$$C_\Pi = \{\leftarrow \texttt{not } m0\_\ell \mid ASP(\Pi) \not\models_C \ell\} \tag{14}$$

Algorithm 3 computes at most $k$ world views of program $\Pi$.

The correctness of Algorithm 3 follows from Proposition 5. We note that the addition of the constraint $C(S)$ to $\Gamma$ (Line 13) is to eliminate any answer set $Y$ of $ASP(\Pi)$ with the property $k1\_m0(S) = k1\_m0(Y)$ and $k0\_m1(S) = k0\_m1(Y)$ from consideration for computing a world view (Line 7) after $S$ has been considered, i.e., this prevents $S$ from being selected repeatedly and thus guarantees that Alg. 3 terminates and computes at most $k$ world views of $\Pi$.

## 3.2 Goal Directed Computing

We now propose a modification of Algorithm 3 that enables the computation of world views satisfying a given objective literal. In other words, given an ELP $\Pi$ and an objective literal $\ell$ in the language of $\Pi$, we would like to compute one (or at most $k$) world view(s) $W$ such that $W \models \text{K}\ell$. This is interesting and has a wide range of applications. For instance, it has been shown that epistemic logic programs can be used for conformant planning (e.g., [Kahl *et al.*, 2015]). It is customary that a conformant planning problem $P$ is translated into an epistemic logic program $E(P)$ with an atom $goal$ for encoding the goal condition such that each world view $W$ of $E(P)$ with $W \models \text{K}goal$ contains a solution to the problem $P$. As such, to compute solutions of $P$, it is sufficient to compute world views satisfying $W \models \text{K}goal$. As it turns out, only a slight modification of Algorithm 3 will be sufficient: replacing **Line 7** of Algorithm 3 with

**Line 7G**: Let $S$ be an answer set of $\Gamma \cup \{\leftarrow \texttt{not } k1\_\ell\}$.

The purpose of the change is to focus on answer set satisfying $k1\_\ell$ of $\Gamma$ which are also answer sets of $ASP(\Pi)$. The new condition requires that $k1\_\ell$ is true in the answer set $S$ which, by the condition (a) in **Step 2** of second section, forces $\ell$ to be in every set of the world view containing $S$.

## 3.3 Adding Heuristics

A further enhancement of the solver can be done by providing it with a way to select a promising answer set that results in the successful identification of a world view. In general, this can be expressed by replacing Line 7 of Algorithm 3 with

**Line 7H**: Let $S$ be an answer set of $\Gamma \cup \{\leftarrow \texttt{not } \ell \mid \ell \in H\}$

where $H$ is a set of literals which could be viewed as a means to represent a heuristic to the solver. Although it might be difficult to identify such heuristics in general, it can be defined for conformant planning. Specifically, we implemented a heuristic for conformant planning that is based on the approach in [Kurien *et al.*, 2002; Nguyen *et al.*, 2012]. Given a conformant planning problem $P$, the system computes solutions of planning problems whose initial states are completions of the initial state of $P$, extracts action occurrences from these solutions to create $H$, and uses this as a heuristic to guide the search.

## 3.4 Implementing a Different Semantics

The semantics of ELPs used in this paper (Section 2) is defined in [Kahl *et al.*, 2015]. It is shown in [Kahl *et al.*, 2016] that with only a slight modification, this semantics is equivalent to the newest semantics defined in [Shen and Eiter, 2016]. Indeed, the newest semantics can be computed by replacing Line 7 of Algorithm 3 with

**Line 7M**: Let $S$ be an answer set of $\Gamma$ such that $\Gamma \cup MC(S)$ does not have an answer set.

where $MC(S)$ is a set of rules aimed at selecting answer set $S'$ of $ASP(\Pi)$ such that $k0\_m1(S) \subset k0\_m1(S')$. More precisely, $MC(S)$ contains (*i*) $r_\alpha$ and $in(\alpha) \leftarrow$ for $\alpha \in k0\_m1(S)$; (*ii*) $ok_m \leftarrow \beta, \texttt{not}\ in(\beta)$ for $\beta$ of the form $k0\_\ell$ or $m1\_\ell$ and $\beta \notin k0\_m1(S)$; and (*iii*) $\leftarrow \texttt{not}\ ok_m$. In the following, we refer to answer sets satisfying the **Line 7M** as those *satisfying the SE-maximality condition*.

## 4 Implementation and Evaluation

We implement the algorithms described in Section 3 in a prototype called EP-ASP and evaluate *(i)* the capabilities of EP-ASP in computing world views of an ELP against `ELPsolve` using the benchmarks provided in [Kahl *et al.*, 2016]; and *(ii)* the effectiveness of directed reasoning of EP-ASP, e.g., in computing world views satisfying a given goal by comparing against DLV-K [Eiter *et al.*, 2003], a logic program based system for conformant planning, using the well-known "bomb in the toilet" problem [Reichgelt, 1987] and its variations. We also implemented a version of EP-ASP that computes the semantics of ELPs given in [Shen and Eiter, 2016] and denote it with EP-ASP$^{se}$.

### 4.1 EP-ASP and EP-ASP$^{se}$

The implementation of EP-ASP and EP-ASP$^{se}$ is in Python. It uses the library provided in the distribution of CLINGO to modify the ASP program and the mode of computation (e.g., cautious or brave reasoning). Its code is omitted for space reason and is downloadable from `https://github.com/tiep/EP-ASP`. EP-ASP$^{se}$ differs from EP-ASP in the implementation of Line 7 (Subsection 3.4). In the following, we write "EP-ASP($^{se}$)" and mean "EP-ASP or EP-ASP$^{se}$."

### 4.2 ELPsolve

`ELPsolve` [Kahl *et al.*, 2016] computes world views of an ELP using semantics given in [Shen and Eiter, 2016]. `ELPsolve` guesses a collection of atoms ($\subseteq$ maximal) of the forms $\texttt{not}$ K$\ell$ or M$\ell$, translates it into a collection of atoms of the form $k1\_\ell$, $k0\_\ell$, $m1\_\ell$, or $m0\_\ell$, adds them to $ASP(\Pi)$, and verifies if the set of answer sets of the new program is a world view.

### 4.3 Experimental Results

All experiments are performed on an Intel Core i7 2.8GHz machine with 16GB memory. Runtimes are reported in second; and '-' indicates that an algorithm fails to solve a problem after one hour. We use an implementation of `ELPsolve` provided by their authors and a publicly-available implementation of DLV-K.[4] Results are detailed in Table 1(a)-(b). All three systems use the program ELPS [Balai and Kahl, 2014] to convert an ELP $\Pi$ to $ASP(\Pi)$.

**EP-ASP($^{se}$) versus** `ELPsolve`: The benchmarks in [Kahl *et al.*, 2016] contain two problems, the *Scholarship Eligibility Problems* [Gelfond, 1991] (represented by E-XX) and *Yale Shooting Problems* [Hanks and McDermott, 1987] (represented by Y-XX). "XX" indicates the number of distinct ground subjective literals in the problem. The ELP encodings of these problems come with the `ELPsolve`'s system.

In E-XX problems, in/eligibility for scholarship of students needs to be determined; and, if it cannot be determined then an interview needs to be scheduled. The difficulty in computing world views for E-XX depends on how many students need to be interviewed. The Y-XX problems are variations of the Yale shooting problem with additional actions and incomplete initial state. Note that in both sets of problems, the semantics in Section 2 is identical to the semantics computed by `ELPsolve`, i.e., all systems compute the same world view.

Table 1(a) details the runtimes of `ELPsolve` and EP-ASP($^{se}$) for computing one world view of E/Y-XX problems. We note that for Y-XX problems, the planning option is turned on for EP-ASP($^{se}$). The number between the brackets in the EP-ASP($^{se}$) column indicates the number of iterations or the number of calls to CLINGO (Lines 7-14, Algorithm 3) that EP-ASP($^{se}$) needs to do to find the first world view. This number for EP-ASP$^{se}$, however, does not include the number of calls to CLINGO before an answer set satisfying the SE-maximality condition described in Subsection 3.4, i.e., checking for maximality of the set of $\texttt{not}$ K$\ell$ and M$\ell$ of the answer set.

We observe that both versions of EP-ASP (**i**) can solve more problems in E-XX problems and (**ii**) consistently faster than `ELPsolve`. In most problems, they are an order of magnitude faster than `ELPsolve`. We believe their better performance lies in the efficient way of identifying candidate world views and its ability to utilize the pre-processor results to prune the search space.

The behavior of EP-ASP$^{se}$ shows that verifying the SE-maximality condition does affect the performance of the system but not significantly (see, E-18/20, E-26, and E-30/32 or Y-20). Its trend seems to be exactly as that of `ELPsolve` as both look for answer sets satisfying the SE-maximality condition before checking for the existence of a world view. On the other hand, EP-ASP behaves a bit different from `ELPsolve`

[4]DLV-K is obtained from `www.dlvsystem.com/dlv/`.

| Prob. | $E^+$ | $S_1$ | $S_2$ | Prob. | $E^+$ | $S_1$ | $S_2$ |
|---|---|---|---|---|---|---|---|
| E-02 | .22 | .02[3] | .01[1] | E-18 | 17.67 | .05[3] | .08[1] |
| E-04 | .23 | .02[2] | .02[1] | E-20 | 37.39 | 5.45[60] | 10.37[23] |
| E-06 | .26 | .02[2] | .02[1] | E-22 | 200.81 | 27.71[126] | 12.55[26] |
| E-08 | .27 | .02[2] | .02[1] | E-24 | 955.88 | 89.64[213] | 12.96[24] |
| E-10 | .28 | .03[3] | .03[1] | E-26 | – | 3.12[39] | 13.07[23] |
| E-12 | .36 | .04[19] | .04[1] | E-28 | – | 30.32[117] | 14.27[20] |
| E-14 | 1.63 | .04[3] | .05[1] | E-30 | – | .14[5] | 16.73[19] |
| E-16 | 6.20 | .05[3] | .06[1] | E-32 | – | .12[3] | 23.37[23] |
| Y-04 | .23 | .02[1] | .03 [1] | Y-17 | .56 | .06[2] | .07 [2] |
| Y-06 | .24 | .03[1] | .03 [1] | Y-20* | 2.37 | 2.32[34] | 4.15 [34] |
| Y-08 | .28 | .05[3] | .06 [3] | Y-23 | 9.88 | .15[4] | .18 [4] |
| Y-10 | .32 | .03[1] | .03 [1] | Y-34 | 419.93 | .05[1] | .06 [1] |

(a) Comparison Between ELPsolve, EP-ASP, and EP-ASP$^{se}$

| Prob. | DLV-K | $S_1$ | $S_2$ | Prob. | DLV-K | $S_1$ | $S_2$ |
|---|---|---|---|---|---|---|---|
| BT (30) | .58 | 1.16(30) | 1.19(30) | BT (90) | 69.27 | 27.89(90) | 28.55(90) |
| BT (50) | 4.63 | 4.82(50) | 4.90(30) | BT (100) | 112.81 | 38.94(100) | 39.92(100) |
| BT (70) | 20.98 | 14.84(70) | 14.27(70) | BT (150) | 813.04 | 167.49(150) | 164.80(150) |
| B1TC (30) | 1.52 | 1.89(60) | 1.95(60) | B1TC (100) | 209.80 | 78.49(200) | 78.32(200) |
| B1TC (50) | 10.38 | 8.41(100) | 8.21(100) | B1TC (150) | 1319.24 | 402.37(300) | 392.94(300) |
| B2TC (3) | .02 | .07(6) | .07(6) | B2TC (10) | – | .32(20) | .35(20) |
| B2TC (5) | .03 | .10(10) | .11(10) | B2TC (30) | – | 4.43(61) | 4.61(61) |
| B2T (7) | 13.06 | .16(14) | .18(14) | B2TC (100) | – | 317.06(200) | 295.08(200) |
| B3TC (3) | .07 | .08(6) | .08(6) | B3TC (10) | – | .51(20) | .47(20) |
| B3TC (5) | .23 | .13(10) | .14(10) | B3TC (30) | – | 7.76(60) | 7.86(60) |
| B3TC (7) | 67.25 | .23(14) | .27(14) | B3TC (50) | – | 41.02(100) | 41.39(100) |
| BTUC (4) | .09 | .30(8) | .39(8) | BTUC (8) | – | 15.93(35) | 23.21(35) |
| BTUC (6) | 256.32 | .73(18) | 1.08(18) | BTUC (10) | – | 37.77(55) | 58.46(55) |

(b) Comparison Between DLV-K, EP-ASP, and EP-ASP$^{se}$

Table 1: Experimental Results ($E^+$: ELPsolve, $S_1$: EP-ASP, $S_2$: EP-ASP$^{se}$; *: problem without world view)

and EP-ASP$^{se}$ when XX increases. EP-ASP guesses an answer set and tries to verify for a world view while the other systems execute the verification only for answer set satisfies the SE-maximality condition. In this sense, EP-ASP uses a *greedy* algorithm whose performance relies on the heuristic of (CLINGO) in computing answer sets. On the other hand, ELPsolve (EP-ASP$^{se}$) systematically searches through all subsets of the set of atoms of the form not K$\ell$ or M$\ell$ (answer sets satisfying the SE-maximality condition).

**EP-ASP versus DLV-K:** We conduct this experiment to evaluate the performance of EP-ASP in computing a world view that satisfies a predefined goal. We compare EP-ASP with DLV-K since DLV-K is one of the most efficient logic programming based implementations for conformant planning. We did not run ELPsolve in this experiment since its current implementation has a limit of 34 subjective literals and this number is exceeded in all problems in this experiment.

The experiments are conducted using the "bomb in the toilet" (BT) problem [Reichgelt, 1987] and its variations, denoted BT($p$), B1TC($p$), B2TC($p$), B3TC($p$), and BTUC($p$) where $p$ indicates the number of packages in a problem. The key idea is that to be safe, one needs to dunk *all* the packets in one of the available toilets (BT). Furthermore, dunking a packet might or might not clog the toilet (C and UC), and one needs to flush the clogged toilet before one can dunk another packet into it. The ELP encoding of these problems are derived from the *world-state encoding* in [Eiter *et al.*, 2003]. We did not use the *knowledge-state encoding* because it uses a different semantics that was not implemented in the generic conformant planning module of EP-ASP($^{se}$).

The runtimes of the systems are detailed in Table 1(b). In this experiment, we use EP-ASP($^{se}$) with the heuristic described in Subsection 3.3. Without the heuristic, EP-ASP($^{se}$) performs significantly slower and does not scale up well. While DLV-K always finds optimal length solution, EP-ASP($^{se}$) does not because of its heuristic. The length of the solution computed by EP-ASP($^{se}$) is given between the parentheses in the column EP-ASP($^{se}$).

We observe that EP-ASP($^{se}$) is able to solve more problems (in B2TC($p$), B3TC($p$), and BTUC($p$) domains) than DLV-K. It is faster than DLV-K when the problem becomes more complex (i.e., increasing $p$) while DLV-K is faster than EP-ASP($^{se}$) for problems with small $p$. The reason is that, in these problems, the times for computing the heuristic dominates the times saved from using the heuristic.

## 5 Conclusions and Future Work

We present an algorithm for computing world views of epistemic logic programs. The new algorithm exploits the relationship between world views and the subjective literals of the program $\Pi$ to avoid the need of computing all answer sets of a modal reduct of $\Pi$ for verifying whether or not a collection of answer sets of the equivalent ASP program $ASP(\Pi)$ is indeed a world view of $\Pi$. We discuss different variations of the algorithm such as the use of heuristics, the computation of world views satisfying a given properties, and the computation of different semantics.

We present two implementations of the algorithm, called EP-ASP and EP-ASP$^{se}$, that take advantage of the multi-shot feature of the answer set solver CLINGO and provide a tight integration of the two processes: world view computation and answer set computation. EP-ASP implements the semantics of ELPs in [Kahl *et al.*, 2015] and EP-ASP$^{se}$ computes the newest semantics of ELPs proposed in [Shen and Eiter, 2016]. Both systems perform exceptionally well comparing to state-of-the-art epistemic logic program solvers in benchmarks from the literature. We also experimentally evaluate the goal directed feature of these systems by comparing it against DLV-K. The experimental results show that they can be competitive with DLV-K in solving conformant planning problems given a good heuristic. It is worth noticing that even though conformant planning has been often mentioned as one of the ideal applications of ELPs, to the best of our knowledge, EP-ASP (or EP-ASP$^{se}$) with heuristic is the first implementation based on ELP that is competitive with DLV-K.

The experimental evaluation shows that epistemic logic program solver can be efficient and could reach the mature level needed to be useful in practical applications that require strong inspection. A particular situation that is of great interest to us is the following problem. Consider an agent who needs to guard a system. The guard knows possible actions that could affect the state of the system. The guard also know that if an enemy **does not know** the truth value of some proposition $p$ then the system is safe. This type of problem could be represented by a conformant planning problem with the goal equivalent to not K$p$. To the best of our knowledge, such problem has not been considered before in the planning community. None of the current conformant planners can solve this type of problem although *all* will solve the problem of finding plans achieving K $p$. This will be our focus in the near future.

## References

[Balai and Kahl, 2014] Evgenii Balai and Patrick Kahl. Epistemic logic programs with sorts. In Daniela Inclezan and Marco Maratea, editors, *ASPOCP 2014*, 2014.

[del Cerro *et al.*, 2015] Luis Farinas del Cerro, Andreas Herzig, and Ezgi Iraz Su. Epistemic equilibrium logic. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015*. AAAI Press / IJCAI, 2015.

[Eiter *et al.*, 2003] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. A logic programming approach to knowledge-state planning, II: the dlv$^{\mathrm{k}}$ system. *Artif. Intell.*, 144(1-2):157–211, 2003.

[Gebser *et al.*, 2014] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–385, 1991.

[Gelfond, 1991] Michael Gelfond. Strong introspection. In Thomas L. Dean and Kathleen McKeown, editors, *AAAI-91*, volume 1, pages 386–391. AAAI Press / The MIT Press, 1991.

[Gelfond, 1994] Michael Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12(1–2):89–116, 1994.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, November 1987.

[Kahl *et al.*, 2015] 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.

[Kahl *et al.*, 2016] Patrick T. Kahl, Anthony P. Leclerc, and Tran Cao Son. A parallel memory-efficient epistemic logic program solver: Harder, better, faster. In *ASPOCP*, 2016.

[Kelly, 2007] Michael Kelly. Wviews: A worldview solver for epistemic logic programs. Honour's thesis, University of Western Sydney, 2007.

[Kurien *et al.*, 2002] James Kurien, P. Pandurang Nayak, and David E. Smith. Fragment-based conformant planning. In *AIPS*, pages 153–162, 2002.

[Lifschitz and Turner, 1994] V. Lifschitz and H. Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proceedings of the Eleventh International Conference on Logic Programming*, pages 23–38, 1994.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

[Nguyen *et al.*, 2012] H-K. Nguyen, D-V. Tran, T.C. Son, and E. Pontelli. On computing conformant plans using classical planners: A generate-and-complete approach. In *Proceedings of the 22st International Conference on Automated Planning and Scheduling, ICAPS 2012,Atibaia, Sao Paulo Brazil, June 25-29*. AAAI Press, 2012.

[Reichgelt, 1987] Han Reichgelt. A review of mcdermott's "critique of pure reason". *AI Commun.*, 0(1):39–42, 1987.

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

[Truszczyński, 2011] Miroslaw Truszczyński. Revisiting epistemic specifications. In Marcello Balduccini and Tran Cao Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*, pages 315–333. Springer, 2011.

[Zhang and Zhao, 2014] Zhizheng Zhang and Kaikai Zhao. ESmodels: An epistemic specification solver. *CoRR*, abs/1405.3486, 2014.

[Zhang, 2006] Yan Zhang. Computational properties of epistemic logic programs. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR-06*, pages 308–317. AAAI Press, 2006.