Epistemic Reasoning in Logic Programs

Yan Zhang
Intelligent Systems Laboratory
School of Computing & Mathematics
University of Western Sydney
Penrith South DC NSW 1797, Australia
E-mail: yan@scm.uws.edu.au

Abstract

Although epistemic logic programming has an enhanced capacity to handle complex incomplete information reasoning and represent agents' epistemic behaviours, it embeds a significantly higher computational complexity than non-disjunctive and disjunctive answer set programming. In this paper, we investigate some important properties of epistemic logic programs. In particular, we show that Lee and Lifschitz's result on loop formulas for disjunctive logic programs can be extended to a special class of epistemic logic programs. We also study the polysize model property for epistemic logic programs. Based on these discoveries, we identify two non-trivial classes of epistemic logic programs whose consistency checking complexity is reduced from PSPACE-complete to NP-complete and Σ_2^P-complete respectively. We observe that many important applications on epistemic representation fall into these two classes of epistemic logic programs.

1 Introduction

As a new logic programming paradigm developed by Gelfond, epistemic logic programming integrates epistemic notions such as knowledge and belief into disjunctive answer set semantics [Gelfond, 1994], which has an enhanced capacity to handle complex incomplete information reasoning and represent agents' epistemic behaviours. However, recently Zhang has shown that epistemic logic programming embeds a significantly higher computational complexity than non-disjunctive and disjunctive answer set programming (assuming that P $\neq$ NP and the polynomial hierarchy does not collapse) [Zhang, 2006]. Therefore, the question of whether there are some *useful* classes of epistemic logic programs that have a lower computational complexity becomes crucial to apply epistemic logic programming to practical domains.

In this paper, we undertake a deep study on this topic. We first refine some key notions of Gelfond's epistemic semantics and show its essential complexity properties. We then show that Lee and Lifschitz's result on loop formulas for disjunctive logic programs can be extended to a special class of epistemic logic programs. We also study the polysize model property for epistemic logic programs. Based on these discoveries, we eventually identify two non-trivial classes of epistemic logic programs whose consistency checking complexity is reduced from PSPACE-complete to NP-complete and Σ_2^P-complete respectively. We observe that many important applications on epistemic representation fall into these two classes of epistemic logic programs.

2 Gelfond's Semantics for Epistemic Reasoning

2.1 Language, Structure and Satisfaction

We consider a language $\mathcal{L}^G$ which is a propositional language augmented with two modal operators K and M to represent knowledge and belief respectively. *Formulas* of $\mathcal{L}^G$ are standard propositional formulas augmented with $K\varphi$ and $M\varphi$ and are closed under $\wedge$, $\vee$ and $\neg$. A *G-structure* $\mathcal{V}$ is a collection of sets of propositional literals called the set of *possible states* of an agent. A *situation* in a G-structure $\mathcal{V}$ is a pair $(\mathcal{V}, s)$, where $s \in \mathcal{V}$. The satisfaction in $\mathcal{L}^G$ is defined in terms of the truth relation and falsity relation between a formula and a situation. The *truth* of a formula φ in a situation $(\mathcal{V}, s)$, denoted as $(\mathcal{V}, s) \models \varphi$, and the *falsity*, denoted as $(\mathcal{V}, s) =| \varphi$, are defined as follows.
$(\mathcal{V}, s) \models p$ iff $p \in s$ where p is a propositional atom.
$(\mathcal{V}, s) \models K\varphi$ iff $(\mathcal{V}, s_i) \models \varphi$ for all $s_i \in \mathcal{V}$.
$(\mathcal{V}, s) \models M\varphi$ iff $(\mathcal{V}, s_i) \models \varphi$ for some $s_i \in \mathcal{V}$.
$(\mathcal{V}, s) \models \varphi \wedge \psi$ iff $(\mathcal{V}, s) \models \varphi$ and $(\mathcal{V}, s) \models \psi$.
$(\mathcal{V}, s) \models \varphi \vee \psi$ iff $(\mathcal{V}, s) \models \varphi$ or $(\mathcal{V}, s) \models \psi$.
$(\mathcal{V}, s) \models \neg\varphi$ iff $(\mathcal{V}, s) =| \varphi$.
$(\mathcal{V}, s) =| p$ iff $\neg p \in s$ where p is an atom.
$(\mathcal{V}, s) =| K\varphi$ iff $(\mathcal{V}, s) \models K\varphi$ does not hold[1].
$(\mathcal{V}, s) =| M\varphi$ iff $(\mathcal{V}, s) \models M\varphi$ does not hold.
$(\mathcal{V}, s) =| \varphi \wedge \psi$ iff $(\mathcal{V}, s) =| \varphi$ or $(\mathcal{V}, s) =| \psi$.
$(\mathcal{V}, s) =| \varphi \vee \psi$ iff $(\mathcal{V}, s) =| \varphi$ and $(\mathcal{V}, s) =| \psi$.

It is clear that if a formula is of the form $K\varphi$, $\neg K\varphi$, $M\varphi$ or $\neg M\varphi$ where φ does not contain operator K or M, that is, φ is an *objective* formula, then its truth value in $(\mathcal{V}, s)$ will not depend on s, and we call this formula *subjective*. In this case, $(\mathcal{V}, s) \models K\varphi$ (or $(\mathcal{V}, s) \models M\varphi$) can be simply written as $\mathcal{V} \models K\varphi$ (or $\mathcal{V} \models M\varphi$ resp.) On the other hand, it is easy to see that an *objective* formula's truth value in $(\mathcal{V}, s)$ will

[1] In this case, we will denote $(\mathcal{V}, s) \not\models K\varphi$.

only depend on s. Therefore, notion $(\mathcal{V}, s) \models \varphi$ may also be simplified as $s \models \varphi$. A propositional literal is also called an *objective literal*, and formulas $KL, \neg KL, ML, \neg ML$ are called *subjective literals* where L is a propositional literal.

It is important to notice that in Gelfond's semantics, $(\mathcal{V}, s) =| \varphi$ is not equivalent to $(\mathcal{V}, s) \not\models \varphi$ in general. This is because by allowing s to be a set of propositional literals, an atom is not assigned a truth value in s if neither of the atom nor its negation is presented in s. Consequently, K and M are not dual modal operators[2] in $\mathcal{L}^G$. For instance, consider $\mathcal{V} = \{\{a, b\}, \{a, b, \neg c\}\}$. Clearly we have $\mathcal{V} \models \neg K \neg c$. But we do not have $\mathcal{V} \models Mc$.

A G-structure $\mathcal{V}$ is called *consistent* if for each $s \in \mathcal{V}$, s does not contain a propositional atom as well as its negation. Otherwise, $\mathcal{V}$ is *inconsistent* and is denoted as $\mathcal{V} = \{Lit\}$ where Lit denotes the inconsistent set of all propositional literals.

Given a G-structure $\mathcal{V}$, we say a formula φ is *satisfied* in $\mathcal{V}$ if there is some $s \in \mathcal{V}$ such that $(\mathcal{V}, s) \models \varphi$. A formula φ is *satisfiable* if φ is satisfied in some consistent G-structure $\mathcal{V}$. A consistent G-structure $\mathcal{V}$ is called a *model* of φ if for every $s \in \mathcal{V}$, $(\mathcal{V}, s) \models \varphi$, denoted as $\mathcal{V} \models \varphi$. We may use $Mod(\varphi)$ to denote the set of all models of φ.

Lemma 1 *A formula φ of $\mathcal{L}^G$ is satisfiable if and only if it is satisfied in a consistent G-structure with at most $|\varphi|$ states, where $|\varphi|$ denotes the length of formula φ.*

Theorem 1 *Deciding whether a formula is satisfiable is NP-complete.*

Given a satisfiable formula φ, very often the maximal model of φ is of a special interest in our study, as will be shown in section 3. Formally a G-structure $\mathcal{V}$ is a *maximal model* of φ if $\mathcal{V}$ is a model of φ and there does not exist another consistent $\mathcal{V}'$ such that $\mathcal{V}' \models \varphi$ and $\mathcal{V} \subset \mathcal{V}'$. Consider formula $K(a \equiv \neg b)$, it has three models $\{\{a, \neg b\}\}$, $\{\{\neg a, b\}\}$ and $\{\{a, \neg b\}, \{\neg a, b\}\}$, where only the last one is a maximal model. On the other hand, formula $((a \wedge \neg b) \equiv \neg Mb) \wedge ((\neg a \wedge b) \equiv \neg Ma)$ has two models $\{\{a, \neg b\}\}$ and $\{\{\neg a, b\}\}$ and both are maximal models.

Proposition 1 *Let φ be a formula of $\mathcal{L}^G$ and let $\mathcal{V}$ be a G-structure. Deciding whether $\mathcal{V}$ is a maximal model of φ is co-NP-complete.*

2.2 World View Semantics for Epistemic Logic Programs

We specify an *epistemic logic program* in language $\mathcal{L}^G$ to be a finite set of rules of the form:

$$F_1 \vee \cdots \vee F_k \leftarrow G_1, \cdots, G_m, \quad not\ G_{m+1}, \cdots, not\ G_n, \tag{1}$$

and constraints of the form:

$$\leftarrow G_1, \cdots, G_m, not\ G_{m+1}, \cdots, not\ G_n. \tag{2}$$

In (1) and (2), $F_1, \cdots, F_k$ are objective literals, $G_1, \cdots, G_m$ are objective or subjective literals, and $G_{m+1}, \cdots, G_n$ are objective literals. Sometimes, we also present a rule in the form $head \leftarrow body$, where $head = \{F_1, \cdots, F_k\}$ and $body = \{G_1, \cdots, G_m, not\ G_{m+1}, \cdots, not\ G_n\}$. Note that if $head = \emptyset$, then the rule becomes a constraint; and if $body = \emptyset$, i.e. $head \leftarrow$, it means that $head$ should be true without any condition.

For an epistemic logic program Π without constraints, its semantics is given by its *world view* which is defined in the following steps:

Step 1. Let Π be an epistemic logic program not containing modal operators K and M and negation as failure not. A set s of propositional literals is called a *belief set* of Π iff s is a minimal set of satisfying conditions: (i) for each rule $F \leftarrow G_1, \cdots, G_m$ from Π such that $s \models G_1 \wedge \cdots \wedge G_m$ we have $s \models F$ (F is $F_1 \vee \cdots \vee F_k$); and (ii) if s contains a pair of complementary literals then we write $s = Lit$.

Step 2. Let Π be an epistemic logic program not containing modal operators K and M and s be a set of propositional literals in the language of Π. By Π_s we denote the result of (i) removing from Π all the rules containing formulas of the form $not\ G$ such that $s \models G$ and (ii) removing from the rules in Π all other occurrences of formulas of the form $not\ G$. We call s is a *belief set* of Π if it is a belief set of Π_s.

Step 3. Finally, let Π be an arbitrary epistemic logic program and $\mathcal{V}$ a collection of sets of propositional literals in its language. By $\Pi_{\mathcal{V}}$ we denote the epistemic logic program obtained from Π by (i) removing from Π all rules containing formulas of the form G such that G is subjective and $\mathcal{V} \not\models G$, and (ii) removing from rules in Π all other occurrences of subjective formulas.

Now we define that a collection $\mathcal{V}$ of sets of ground literals is a *world view* of Π if $\mathcal{V}$ is the collection of all belief sets of $\Pi_{\mathcal{V}}$. Clearly, a world view of an epistemic logic program is also a G-structure in language $\mathcal{L}^G$.

It is easy to extend the world view definition above for epistemic logic programs Π with constraints: $\mathcal{V}$ is a world view of Π if $\mathcal{V}$ is a world view of the program obtained by deleting all constraints from Π, and for each constraint in Π of the form (2), either $\mathcal{V} \not\models G_i$ for some $1 \leq i \leq m$, or $\mathcal{V} \models G_j$ for some $(m+1) \leq j \leq n$. An epistemic logic program may have one, more than one, or no world views.

Theorem 2 *[Zhang, 2006] Deciding whether an epistemic logic program has a world view is PSPACE-complete.*

It is easy to see that extended/normal logic programs and disjunctive logic programs are two special classes of epistemic logic programs whose world views always exist. Although computing the world view (i.e. the collection of all stable models/answer sets) for these types of programs could be difficult (this will be addressed in section 7), in general, we are more interested in discovering non-trivial classes of epistemic logic programs with a lower computational complexity.

3 Completion for Epistemic Logic Programs without Positive Knowledge

Let us consider a simple epistemic logic program $\Pi = \{a \leftarrow \neg Mb\}$. This program has one world view $\{\{a\}\}$. However,

[2] K and M are called *dual* if $\neg K \neg \varphi$ is equivalent to $M\varphi$.

under the previous G-structure semantics, Π's equivalent formula $\neg b \wedge (a \equiv \neg Mb)$[3] has a unique model $\{\{a, \neg b\}\}$, which is obviously different from Π's world view. This inconsistency can be avoided if we slightly revise the definitions of G-structure and the associated satisfaction in $\mathcal{L}^G$.

To begin with, we first need to remove negative propositional atoms from epistemic logic programs. An epistemic logic program Π is said in *canonical form* if no negative propositional atom occurs in Π. Note that this does not exclude occurrences of negative subjective formulas in Π. For instance, program $\{a \vee b \leftarrow \neg Mc\}$ is to be in canonical form, while program $\{a \vee b \leftarrow \neg M \neg c\}$ is not. For an arbitrary program Π, we can always transform it into the canonical form by introducing new propositional atoms in the language. In particular, for each atom a in $\mathcal{L}^G$, we introduce a new atom a' to represent literal $\neg a$. Therefore, by replacing each negative atom $\neg a$ occurring in Π with the corresponding new atom a', the resulting program, denoted as Π^c, is in canonical form. If $\mathcal{V}$ is a G-structure in language $\mathcal{L}^G$, $\mathcal{V}^c$ is obtained by replacing each negative atom occurring in $\mathcal{V}$ with the corresponding new atom.

Proposition 2 *Let Π be an epistemic logic program and Π^c be Π's canonical form. A consistent G-structure $\mathcal{V}$ is a world view of Π if and only if $\mathcal{V}^c$ is a world view of Π^c.*

Due to Proposition 2, in the rest of this paper, we can assume that all epistemic logic programs are in canonical form in our context. Now we re-define a G-structure to be a collection of sets of propositional atoms. Consequently, we revise the falsity relation in $\mathcal{L}^G$ as follows: $(\mathcal{V}, s) =| \neg p$ iff $p \notin s$ (p is a propositional atom), while the truth relation and falsity relation for all other formulas remain the same. We should mention that this change will imply that K and M become dual modal operators in $\mathcal{L}^G$ (recall that in section 2.1, we showed that K and M are not dual operators). Despite this difference, this new satisfaction will not affect the world view definition for epistemic logic programs.

Given an epistemic logic program Π, we use $Atom(\Pi)$ to denote the set of all atoms occurring in Π. Now we define a class of epistemic logic programs without positive knowledge, denoted as $\mathcal{ELP}^{-K}$. That is, for each $\Pi \in \mathcal{ELP}^{-K}$, no knowledge operator K *positively* occurs in the body of each rule. It is easy to obverse that many typical epistemic logic programs in applications are actually $\mathcal{ELP}^{-K}$ programs [Gelfond, 1994].

Definition 1 *Let Π be an $\mathcal{ELP}^{-K}$ program. The* completion *of Π, denoted as $Comp(\Pi)$, consists of the following formulas: (1) if $head \leftarrow body$ is a rule in Π, then the following formula is in $Comp(\Pi)$*[4]*:*

$$body \supset head; \tag{3}$$

(2) for each $a \in Atom(\Pi)$, the following formula is in $Comp(\Pi)$:

$$a \supset \bigvee_{head \leftarrow body \in \Pi, a \in head} (body \wedge \bigwedge_{p \in head \setminus \{a\}} \neg p); \tag{4}$$

and (3) for each $a \in Atom(\Pi)$, if there is no rule $head \leftarrow body$ in Π such that $a \in head$, then $\neg a$ is in $Comp(\Pi)$.

Example 1 Let Π consist of the following rules:

$a \vee b \leftarrow \neg Mc,$
$d \leftarrow \neg Ka.$

Clearly, Π is an $\mathcal{ELP}^{-K}$ program. From Definition 1, $Comp(\Pi)$ consists of the following formulas:

$\neg Mc \supset (a \vee b),$
$a \supset (\neg b \wedge \neg Mc),$
$b \supset (\neg a \wedge \neg Mc),$
$d \equiv \neg Ka,$
$\neg c.$

It is easy to check that Π has a unique world view $\{\{a, d\}, \{b, d\}\}$, which is also the unique model of $Comp(\Pi)$. □

In general, Definition 1 cannot be extended to arbitrary epistemic logic programs. Consider the following program Π':

$a \vee b \leftarrow \neg Mc,$
$d \leftarrow Ka.$

If we apply Definition 1 to Π', its completion is the same as in Example 1 except formula $d \equiv \neg Ka$ is replaced by $d \equiv Ka$. We can see that Π' has one world view $\{\{a\}, \{b\}\}$, where $Comp(\Pi')$ has two maximal models $\{\{a\}, \{b\}\}$ and $\{\{a, d\}\}$. Since Π' does not contain any loop (see next section), Definition 1 actually does not provide a precise completion characterization for arbitrary epistemic logic programs.

The following proposition establishes an important connection between an $\mathcal{ELP}^{-K}$ program and its completion.

Proposition 3 *Let Π be an $\mathcal{ELP}^{-K}$ program. If $\mathcal{V}$ is a world view of Π, then it is also a maximal model of $Comp(\Pi)$.*

4 Loop Formulas for $\mathcal{ELP}^{-K}$ Programs

Now we show that Lee and Lifschitz's loop formula result for disjunctive logic programs [Lee and Lifschitz, 2003] can also be extended to $\mathcal{ELP}^{-K}$ programs.

Given an epistemic logic program Π, the *positive dependency graph* of Π, denoted as G^+_Π, is the directed graph constructed in the following way: the set of vertices is $Atom(\Pi) \cup KM(\Pi)$, where $KM(\Pi)$ is the set of all subjective atoms occurring in Π, and for each pair of vertices $x, y \in Atom(\Pi) \cup KM(\Pi)$: (1) there is an edge from x to y if there is a rule $head \leftarrow body$ in Π, such that $x \in head$, and y *positively* occurs in $body$; (2) for each path in the graph formed from (1), if both a and Ka (or Ma) occur in the path, then for any $x', y' \in Atom(\Pi) \cup KM(\Pi)$ such that (x', a) and (a, y') are edges, add edges (x', Ka) and (Ka, y') ((x', Ma) and (Ma, y') resp.) respectively; (3) based the graph formed from (1) and (2), if there is no such cycle containing vertex a but not Ka (Ma resp.), then remove vertex a and all its associated edges. We may use $Pos(body)$ to denote the set of all (objective and subjective) atoms positively occurring in $body$. We call a nonempty set $L \subseteq Atom(\Pi) \cup KM(\Pi)$ a *loop* of Π if for any x and y in L, there is a path in G^+_Π from x to y with length > 0.

[3] Later we will see that this formula is the completion of Π.

[4] When we view *body* as a formula in language $\mathcal{L}^G$, the negation as failure *not* in *body* is interpreted as the classical negation $\neg$.

Definition 2 *Let* Π *be an epistemic logic program and* L *a loop of* Π. $R(L)$ *is denoted as the set of following formulas:*

$$body \wedge \bigwedge_{p \in head \setminus L} \neg p,$$

for each rule $head \leftarrow body$ *in* Π *such that* $head \cap L \neq \emptyset$ *and* $Pos(body) \cap L = \emptyset$. *Then the* loop formula $LF(L)$ *associated with* L *in* Π *is specified as*

$$(\bigvee L) \supset (\bigvee R(L)). \quad (5)$$

We use $LF(\Pi)$ *to denote the set of all loop formulas in* Π.

It is easy to see that (5) is equivalent to Lee and Lifschitz's disjunctive loop formula [Lee and Lifschitz, 2003] and Lin and Zhao's loop formula [Lin and Zhao, 2004] if Π is restricted to a disjunctive logic program and a normal logic program respectively.

Example 2 Consider an epistemic logic program Π:

$a \vee b \leftarrow Mc,$
$c \leftarrow a,$
$d \leftarrow \neg Ka.$

Π has one loop $L = \{a, Mc\}$. Then we have $R(L) = \emptyset$, and $LF(L) = \{\neg(a \vee Mc)\}$. On the other hand, since Π is an $\mathcal{ELP}^{-K}$, Π's completion $Comp(\Pi)$ consists of the following formulas:

$Mc \supset a \vee b,$
$a \supset ((\neg b \wedge Mc) \vee c),$
$b \supset (\neg a \wedge Mc),$
$c \supset a,$
$d \equiv \neg Ka.$

We can see that $Comp(\Pi)$ has two (maximal) models $\{\{a, c\}\}$ and $\{\{d\}\}$, where $Comp(\Pi) \cup LF(L)$ has one (maximal) model $\{\{d\}\}$, which is also the unique world view of Π. $\square$

Theorem 3 *Let* Π *be an* $\mathcal{ELP}^{-K}$ *program,* $Comp(\Pi)$ *the completion of* Π*, and* $LF(\Pi)$ *the set of loop formulas of* Π. *A* G*-structure is a world view of* Π *if and only if it is a maximal model of* $Comp(\Pi) \cup LF(\Pi)$.

4.1 Proof of Theorem 3

In order to prove Theorem 3, we need to first show some important properties of formulas and epistemic logic programs in language $\mathcal{L}^G$. We first introduce the notion of epistemic reduction for formulas.

Let $\mathcal{V}$ be a G-structure and φ a formula of $\mathcal{L}^G$. A formula $\varphi^{\mathcal{V}}$ is called the *epistemic reduction* of φ with respect to $\mathcal{V}$ if $\varphi^{\mathcal{V}}$ is obtained from φ in the following way: (1) for each φ's subformula of the form $K\phi$ (here ϕ is objective), $K\phi$ is replaced by ϕ if $\mathcal{V} \models K\phi$, otherwise $K\phi$ is replaced by $\mathtt{F}$; and (2) for each φ's subformula of the form $M\phi$, $M\phi$ is replaced by $\mathtt{T}$ if $\mathcal{V} \models M\phi$, otherwise $M\phi$ is replaced by $\mathtt{F}$.

Consider formula $\varphi = Ka \vee Mb$. Suppose a and b are the only propositional atoms in the language. Then $\mathcal{V} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ is a maximal model of φ. Furthermore, we can see that $\varphi^{\mathcal{V}} = \mathtt{T}$, where $\mathcal{V}$ is exactly the set of all models of $\varphi^{\mathcal{V}}$.

Proposition 4 *Let* φ *be a formula of* $\mathcal{L}^G$ *and let* $\mathcal{V}$ *be a* G*-structure. Then* $\mathcal{V}$ *is a maximal model of* φ *if and only if* $Mod(\varphi^{\mathcal{V}}) = \{s \mid s \in \mathcal{V}\}$.

Now with a slight change, we may extend the notion of epistemic reduction to epistemic logic programs. Let Π be an epistemic logic programs and $\mathcal{V}$ be a G-structure in language $\mathcal{L}^G$. A disjunctive logic program (without containing any K and M operators) $eReduct(\Pi, \mathcal{V})$ is called the *epistemic reduction* of Π under $\mathcal{V}$ if it is obtained from Π in the following way: (1) removing all rules from Π where G is a subjective literal occurring in the rules and $\mathcal{V} \not\models G$, and (2) removing all other occurrences of subjective literals in the remaining rules (i.e. replacing those G with $\mathtt{T}$ due to $\mathcal{V} \models G$). Then we have the following results.

Proposition 5 *Let* Π *be an epistemic logic program and* $\mathcal{V}$ *a* G*-structure of* $\mathcal{L}^G$. $\mathcal{V}$ *is a world view of* Π *if and only if* $\mathcal{V}$ *is the collection of all stable models of disjunctive logic program* $eReduct(\Pi, \mathcal{V})$.

The following two lemmas establish an important connection between an $\mathcal{ELP}^{-K}$ program and its completion and loop formulas.

Lemma 2 *Let* Π *be an* $\mathcal{ELP}^{-K}$ *program and* $\mathcal{V}$ *a world view of* Π. *Then* $(Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}} \equiv Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$.

Lemma 3 *Let* Π *be an* $\mathcal{ELP}^{-K}$ *program and* $\mathcal{V}$ *a maximal model of* $Comp(\Pi) \cup LF(\Pi)$. *Then* $(Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}} \equiv Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$.

Proof of Theorem 3: Consider an $\mathcal{ELP}^{-K}$ program Π and $Comp(\Pi) \cup LF(\Pi)$. We show that $\mathcal{V}$ is a world view of Π iff $\mathcal{V}$ is a maximal model of $Comp(\Pi) \cup LF(\Pi)$.

($\Rightarrow$) Suppose $\mathcal{V}$ is a world view of Π. Then from Proposition 5, we know that $\mathcal{V}$ is the collection of all stable models of disjunctive program $eReduct(\Pi, \mathcal{V})$. On the other hand, from Lemma 2, we have $(Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}} \equiv Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$. On the other hand, from Theorem 1 in [Lee and Lifschitz, 2003], we know that each stable model of program $eReduct(\Pi, \mathcal{V})$ is also a model of formula $Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$. So we conclude $\mathcal{V} = Mod((Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}})$. Finally, from Proposition 4, it follows that $\mathcal{V}$ is a maximal model of $Comp(\Pi) \cup LF(\Pi)$.

($\Leftarrow$) Suppose $\mathcal{V}$ is a maximal model of $Comp(\Pi) \cup LF(\Pi)$. From Lemma 3, we know $(Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}} \equiv Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$. According to Proposition 4, on the other hand, $\mathcal{V} = Mod((Comp(\Pi) \cup LF(\Pi))^{\mathcal{V}})$. Then from Theorem 1 in [Lee and Lifschitz, 2003], each model of $Comp(eReduct(\Pi, \mathcal{V})) \cup LF(eReduct(\Pi, \mathcal{V}))$ is also a stable model of disjunctive logic program $eReduct(\Pi, \mathcal{V})$. Finally, from Proposition 5, it follows that $\mathcal{V}$ is a world view of Π. This completes our proof of Theorem 3. $\square$

5 Polynomial Bound on Loop Formulas

Since there may be an exponential number of loops in an epistemic logic program, the way of transforming an $\mathcal{ELP}^{-K}$

program into $Comp(\Pi) \cup LF(\Pi)$ and then computing its G-structures will not have particular computation advantages. However, we will show that for $\mathcal{ELP}^{-K}$ programs with polynomially bounded numbers of loop formulas, the consistency check problem will be significantly reduced from PSPACE-complete to NP-complete. Formally, given an epistemic logic program Π, we say that Π has a *polynomial bound* on its loop formulas if the number of cycles in G^+_Π is bounded by $\mathcal{O}(f(|G^+_\Pi|))$ where $f(|G^+_\Pi|)$ is a polynomial of $|G^+_\Pi|$.

Theorem 4 *Let Π be an $\mathcal{ELP}^{-K}$ program that has a polynomial bound on its loop formulas. Deciding whether Π has a world view is NP-complete.*

Proof: Here we only give membership proof. From Theorem 3, Π has a world view iff $Comp(\Pi) \cup LF(\Pi)$ has a maximal model. Clearly, if the number of cycles in G^+_Π is bounded by $\mathcal{O}(f(|G^+_\Pi|))$, then the size of formula $Comp(\Pi) \cup LF(\Pi)$ is also bounded by $\mathcal{O}(f(|G^+_\Pi|))$. So we can guess such $LF(\Pi)$ in polynomial time with a nondeterministic Turing machine. From Theorem 1, we know that checking whether $Comp(\Pi) \cup LF(\Pi)$ has a model is in NP. Also it is easy to see that $Comp(\Pi) \cup LF(\Pi)$ has a model iff it has a maximal model. □

In the following, we provide a polynomial upper bound on the number of loop formulas for an epistemic logic program. Given an epistemic logic program Π and $a \in head(\Pi)$, an *inference chain* of a is a sequence of atoms $(a, b_1, \cdots, b_{l-1}, b_l)$, where $a, b_1, \cdots, b_l$ are different atoms in $Atom(\Pi)$, and there are l different rules $r_1, \cdots, r_l$ in Π such that $a \in head(r_1), b_1 \in Pos(body(r_1))$, $b_1 \in head(r_2)$, $b_2 \in Pos(body(r_2)), \cdots, b_{l-1} \in head(r_l)$ and $b_l \in Pos(body(r_l))$. For each $a \in head(\Pi)$, it may have one or more inference chains. The *inference depth* of a, denoted as $i\text{-}depth(a)$, is the length of a's longest inference chain. The *inference depth* of Π is defined as $i\text{-}depth(\Pi) = max(i\text{-}depth(a) : a \in head(\Pi))$.

Proposition 6 *Let Π be an epistemic logic program. If $i\text{-}depth(\Pi) < k$ for some fixed k, then $|LF(\Pi)|$ is bounded by $\mathcal{O}((2^k/k!) \cdot n^k)$, where $|Atom(\Pi)| = n$.*

Proposition 7 *Let Π be an epistemic logic program. For a fixed k, deciding whether $i\text{-}depth(\Pi) < k$ is solvable in time $\mathcal{O}(k \cdot n^{k+1})$ where $|Atom(\Pi)| = n$.*

6 Polysize Model Property

Some special modal logics such as single agent S5 and KD45 [Blackburn *et al.*, 2001] have *polysize model property* which brings the consistency check (satisfiability problem) in these logics down to NP-complete. We have shown that Gelfond's epistemic semantics also has this property (i.e. Lemma 1). However, epistemic logic programs under the world view semantics do not satisfy the polysize model property in general. For instance, the following program

$x_i \vee x'_i \leftarrow,$
$y_i \leftarrow \neg M y'_i,$
$y'_i \leftarrow \neg M y_i,$ where $i = 1, \cdots, n,$

has an exponential number of world views and each world view contains an exponential number of belief sets. Nevertheless, it is easy to observe that many specific epistemic logic programs indeed satisfy the polysize model property. Consequently, we can further identify an important class of epistemic logic programs whose consistency check is Σ^P_2-complete.

Definition 3 *An epistemic logic program Π is said to satisfy the* polysize model property *if either Π has no world view, or for each Π's world view $\mathcal{V}$, $|\mathcal{V}|$ is bounded by $\mathcal{O}(f(|\Pi|))$, where $f(|\Pi|)$ is a polynomial of $|\Pi|$.*

Theorem 5 *Let Π be an epistemic logic program satisfying the polysize model property. Deciding whether Π has a world view is Σ^P_2-complete.*

Proof: Since for each world view $\mathcal{V}$ of Π, $|\mathcal{V}|$ is bounded by $\mathcal{O}(f(|\Pi|))$, we can guess, in polynomial time, a k with k being bounded by $\mathcal{O}(f(|\Pi|))$ and a collection $\mathcal{V}$ consisting of k different sets of atoms. Then we check whether $\mathcal{V}$ is a world view of Π in the following way: (1) transform Π into a disjunctive logic program $\Pi_\mathcal{V}$ by performing Step 3 in section 2.2, and (2) for each $s \in \mathcal{V}$, check whether s is a stable model of $\Pi_\mathcal{V}$. Clearly, (1) can be done in polynomial time, and (2) is solvable with k queries to an NP oracle.

Hardness proof. The hardness is proved by a reduction of the validity of QBF $A = \exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m A'$ $(n, m \geq 1)$, where $A' = D_1 \vee \cdots \vee D_h$ and each $D_k = l_{k,1} \wedge l_{k,2} \wedge l_{k,3}$ is a conjunction of propositional literals $l_{i,j}$ over $\{x_1, \cdots, x_n, y_1, \cdots, y_m\}$. We know that deciding whether A is valid is Σ^P_2-complete [Papadimitriou, 1995]. We construct in polynomial time an epistemic logic program Π from A and show that Π has a world view and each of its world views contains exactly one belief set if and only if A is valid. In particular, Π is constructed on propositional atoms $\{x_1, \cdots, x_n, y_1, \cdots, y_m\} \cup \{x'_1, \cdots, x'_n, y'_1, \cdots, y'_m\} \cup \{valid, invalid\}$, where x' and y' are used to imitate the negations of x and y respectively. Π consists of the following rules:

$x_i \leftarrow \neg M x'_i,$
$x'_i \leftarrow \neg M x_i,$ for $i = 1, \cdots, n,$
$y_j \vee y'_j \leftarrow,$
$y_j \leftarrow K valid,$
$y'_j \leftarrow K valid,$ for $j = 1, \cdots, m,$
$valid \leftarrow \rho(l_{k,1}), \rho(l_{k,2}), \rho(l_{k,3}),$ for $k = 1, \cdots, h,$
$invalid \leftarrow \neg M invalid, not\ valid,$

where $\rho(l) = a$ if $l = a$ and $\rho(l) = a'$ if $l = \neg a$ for any $a \in \{x_1, \cdots, x_n, y_1, \cdots, y_m\}$. It is easy to see that if Π has a world view $\mathcal{V}$, then $\mathcal{V}$ must only contain *one* belief set. Then we can show that Π has a world view if and only if A is valid. □

Now we investigate under what conditions, an epistemic logic program will satisfy the polysize model property. We first define the dependency graph for an epistemic logic program. The *dependency graph* G_Π of Π is a directed graph where the set of vertices is $Atom(\Pi)$; and for vertices a and b, there is an edge (a, b) from a to b if there is a rule

$head \leftarrow body$ in Π such that $a \in head$ and $b \in body$ or $a, b \in head$ ($a \neq b$). The edge (a, b) is labeled with $+$ if b positively occurs in $body$; (a, b) is labeled with $-$ if $not\ b$ occurs in $body$ or b occurs in $head$ (assuming $a \neq b$); and (a, b) is labeled with $-K$ or $-M$ if $\neg Kb$ or $\neg Mb$ occurs in $body$ respectively. A cycle in G_Π is called *negative cycle* if the cycle has at least one edge labeled with $-$ while all other edges in the cycle (if there are some) are labeled with $+$. A cycle is called a *mixed negative cycle* if the cycle has at least one edge labeled with $-K$ or $-M$.

Proposition 8 *Let Π be an epistemic logic program, G_Π its dependency graph and k a fixed number. Then the following results hold:*

1. *If G_Π has k negative even cycles*[5] *and Π has a world view, then Π's each world view contains at most 2^k belief sets;*
2. *If G_Π has k mixed negative even cycles, then Π has at most 2^k world views;*
3. *If G_Π has k negative even cycles but has no mixed negative even cycle, then either Π has no world view or has a unique world view containing at most 2^k belief sets.*

Corollary 1 *For any epistemic logic program Π where its dependency graph G_Π has a fixed number of negative even cycles, deciding whether Π has a world view is Σ_2^P-complete.*

7 Concluding Remarks

As argued by Gelfond, epistemic logic programming has a rich expressive power for epistemic representation which other modal nonmonotonic logics do not have. Nevertheless, as shown in [Zhang, 2006], the enhanced expressive power is due to a higher computational cost than other nonmonotonic logics and logic programming paradigms. Although this is the case in general, as we will observe next, many important applications of epistemic representation actually fall into the two classes of epistemic logic programs that we identified earlier.

One important application of epistemic logic programming is to represent the *epistemic closed world assumption* (ECWA) which involves a strong introspection. It was noted that other logic programs and modal nonmonotonic logics such as autoepistemic logic [Marek and Truszczynski, 1991] usually have difficulties to handle it. As demonstrated in [Gelfond, 1994], representing ECWA is important when disjunctive information is presented in the agent's knowledge base. In epistemic logic programs, ECWA can be simply represented by rules like

$$a' \leftarrow \neg Ma,$$

for all atoms in the language[6]. We observe that adding such ECWA rules into an epistemic logic program neither introduces any new loop formulas nor increases the size of world views of the program, although they may generate more world views. Therefore, in principle ECWA rules do not add an extra burden on the world view computation for an epistemic logic program.

Now let us examine the following normal logic program Π^* given in [Lifschitz and Razborov, 2006]:

$$p_{ij} \leftarrow not\ q_{ij},$$
$$q_{ij} \leftarrow not\ p_{ij},$$
$$r_1 \leftarrow,$$
$$r_j \leftarrow r_i, p_{ij}, i, j = 1, \cdots, n.$$

If we view Π^* as an epistemic logic program, then this program does not belong to either of the two classes we identified earlier. Computing Π^*'s world view would be hard because Π^* has 2^n loop formulas and the unique world view also contains 2^{n^2} belief sets.

Program Π^* defines the reachability in arbitrary directed graphs. Finding a reachable path from r_1 to r_i is to compute *one* answer set containing r_i. When we review Π^* as an epistemic logic program, we have to compute the world view that is the collection of all answer sets of Π^*. This actually can be avoided if we rewrite Π^* to $\Pi^\dagger$ as follows:

$$p_{ij} \leftarrow \neg Mq_{ij},$$
$$q_{ij} \leftarrow \neg Mp_{ij},$$
$$r_1 \leftarrow,$$
$$r_j \leftarrow r_i, p_{ij}, i, j = 1, \cdots, n.$$

Program $\Pi^\dagger$ has an exponential number of world views but each one only contains one belief set of size $2n$. Then under program $\Pi^\dagger$, to find a reachable path from r_1 to r_i, we only need to compute one world view of $\Pi^\dagger$ where its belief set contains r_i. In this way, the computation is the same as that of program Π^* under the answer set semantics.

[5] A cycle contains an even number of edges labeled with $-$.

[6] Here a' is used to represent the negation of atom a.

References

[Blackburn *et al.*, 2001] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[Gelfond, 1994] M. Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12:98–116, 1994.

[Lee and Lifschitz, 2003] J. Lee and V. Lifschitz. Loop formulas for disjunctive logic programs. In *Proceedings of ICLP-2003*, pages 451–465, 2003.

[Lifschitz and Razborov, 2006] V. Lifschitz and A. Razborov. Why are there so many loop formulas? *ACM Transaction on Computational Logic*, 7:261–268, 2006.

[Lin and Zhao, 2004] F. Lin and Y. Zhao. Assat: computing answer sets of a logic program by sat solvers. *Artificial Intelligence*, 157:115–137, 2004.

[Marek and Truszczynski, 1991] W. Marek and M. Truszczynski. Autoepistemic logic. *Journal of ACM*, 38:588–619, 1991.

[Papadimitriou, 1995] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1995.

[Zhang, 2006] Y. Zhang. Computational properties of epistemic logic programs. In *Proceedings of KR-2006*, pages 308–317, 2006.