Computational Properties of Epistemic Logic Programs

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

Abstract

Gelfond's epistemic logic programs are not only an extension of disjunctive extended logic programs for handling difficulties in reasoning with incomplete information, but also an effective formalism to represent agents' epistemic reasoning under a logic programming setting. Recently there is an increasing research in this direction. However, for many years the complexity of epistemic logic programs remains unclear. This paper provides a precise answer to this problem. We prove that consistency check for epistemic logic programs is in PSPACE and this upper bound is also tight. The approach developed in our proof is of interest on its own: it immediately yields an algorithm to compute world views of an epistemic logic program, and it can also be used to study computational properties of nested epistemic logic programs - a significant generalization of Gelfond's formalism.

Introduction

Gelfond's epistemic logic programs are an extension of disjunctive extended logic programs to overcome the difficulty in reasoning about incomplete information with the present of multiple extensions (Gelfond 1994). Since epistemic logic programs integrate the answer set semantics and classical Kripke possible worlds semantics on knowledge and belief, they are also an effective formalism to represent agents' epistemic reasoning under a logic programming setting. Recently there is an increasing research in this direction, e.g. (Lobo, Mendez, & Taylor 2001; Wang & Zhang 2005; Watson 2000; Zhang 2003).

However, for many years the complexity of epistemic logic programs remains unclear. For instance, it is unknown what is the complexity class that consistency check for epistemic logic programs belongs to. The main difficulty in this study seems that many common techniques developed in complexity study for logic programming and modal logics appear hard to apply in the case of epistemic logic programs.

Let us take a closer look at this aspect. As each disjunctive extended logic program is a special epistemic logic program, we immediately conclude that consistency check for epistemic logic programs is Σ_2^P-hard (Eiter & Gottlob 1995). But this lower bound seems not tight, because it is easy to think of an epistemic logic program that has an exponential number of world views (models) and each world view contains an exponential number of belief sets. For example, the following program is of this property:

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

A straightforward checking on the existence of the world view of an arbitrary epistemic logic program would take exponential time and use exponential space. Then we may consider to employ some traditional complexity proof techniques in modal logics. But this seems difficult too. First, our above sample program shows that epistemic logic programs do not have a *polysize model property* like some modal logics have (e.g. single agent S5 modal logic) (Halpern & Moses 1992). Second, it will not be feasible to find an efficient way to transform an epistemic logic program into some standard modal logic formula and then apply existing techniques suitable in that modal logic, because as pointed by Lifschitz and Razborov in (Lifschitz & Razborov 2006), under answer set semantics, it is unlikely to have any transformation that ensures a polysize translated formula generally. Finally we observe that other well known proof techniques for modal logics such as Witness and Tableau methods (Blackburn, de Rijke, & Venema 2001) cannot be directly applicable to epistemic logic programs due to their nonmonotonicity.

Therefore, it is clear that we need some new approach to study the complexity of epistemic logic programs. In this paper, we provide a precise answer to the complexity problem of epistemic logic programs. We prove that consistency check for epistemic logic programs is in PSPACE and this upper bound is also tight. The key technique used in our proof is: instead of checking a collection of belief sets, as the world view semantics defines and obviously which needs exponential space, we generate every epistemic valuation one by one which reduces an epistemic logic program into a disjunctive extended logic program, and then we generate all answer sets of this disjunctive extended logic program, also one at a time, to check if such epistemic valuation is coherent with the original program. The generation of epistemic valuations becomes automatic by encoding such valuations as binary strings which are of polynomial sizes, and the whole process only uses polynomial space although it runs in exponential time. The approach developed in our proof is of interest on its own: it immediately yields an algorithm to compute world views of an epistemic logic program, and it can also be used to study computational properties of nested epistemic logic programs - a generalization of Gelfond's epistemic logic programs with a richer expressive power (Wang & Zhang 2005).

The paper is organized as follows. Section 2 gives an overview on epistemic logic programs. Section 3 describes the main complexity results with details of the upper bound proof for the consistency check problem. Based on the approach developed in section 3, section 4 proposes an algorithm to compute world views of an epistemic logic program, and considers under what conditions the algorithm can be improved. Section 5 presents the complexity results for nested epistemic logic programs. Finally section 6 concludes this paper with some remarks.

Epistemic Logic Programs: An Overview

In this section, we present a general overview on epistemic logic programs. Gelfond extended the syntax and semantics of disjunctive extended logic programs to allow the correct representation of incomplete information (knowledge) in the presence of multiple extensions (Gelfond 1994). Consider the following disjunctive program about the policy of offering scholarships in some university[1]:

Π:
r_1: $eligible(x) \leftarrow highGPA(x)$,
r_2: $eligible(x) \leftarrow minority(x), fairGPA(x)$,
r_3: $\neg eligible(x) \leftarrow \neg fairGPA(x),$
$\qquad \neg highGPA(x)$,
r_4: $interview(x) \leftarrow not\ eligible(x),$
$\qquad not\ \neg eligible(x)$,
r_5: $fairGPA(mike); highGPA(mike) \leftarrow$,

while rule r_4 can be viewed as a formalization of the statement: "the students whose eligibility is not decided by rules r_1, r_2 and r_3 should be interviewed by the committee". It is easy to see that $\mathcal{P}$ has two answer sets $\{highGPA(mike),\ eligible(mike)\}$ and $\{fairGPA(mike),\ interview(mike)\}$. Therefore the answer to query $interview(mike)$ is *unknown*, which seems too weak from our intuition. Epistemic logic programs will overcome this kind of difficulties in reasoning with incomplete information.

In epistemic logic programs, the language of (disjunctive) extended logic programs is expanded with two modal operators K and M. KF is read as "F is known to be true" and MF is read as "F may be believed to be true". For our purpose, in this paper we will only consider propositional epistemic logic programs where rules containing variables are viewed as the set of all ground rules by replacing these variables with all constants occurring in the language. The semantics for epistemic logic programs is defined by pairs $(\mathcal{A}, W)$, where $\mathcal{A}$ is a collection of sets of ground literals called the set of *possible beliefs* of certain agent, while W is a set in $\mathcal{A}$ called the agent's *working set of beliefs*. The truth of a formula F in $(\mathcal{A}, W)$ is denoted by $(\mathcal{A}, W) \models F$ and the falsity by $(\mathcal{A}, W) =\!| F$, and are defined as follows.
$(\mathcal{A}, W) \models F$ iff $F \in W$ where F is a ground atom.
$(\mathcal{A}, W) \models KF$ iff $(\mathcal{A}, W_i) \models F$ for all $W_i \in \mathcal{A}$.
$(\mathcal{A}, W) \models MF$ iff $(\mathcal{A}, W_i) \models F$ for some $W_i \in \mathcal{A}$.
$(\mathcal{A}, W) \models F \wedge G$ iff $(\mathcal{A}, W) \models F$ and $(\mathcal{A}, W) \models G$.
$(\mathcal{A}, W) \models F; G$ iff $(\mathcal{A}, W) \models \neg(\neg F \wedge \neg G)$.
$(\mathcal{A}, W) \models \neg F$ iff $(\mathcal{A}, W) =\!| F$.
$(\mathcal{A}, W) =\!| F$ iff $\neg F \in W$ where F is a ground atom.
$(\mathcal{A}, W) =\!| KF$ iff $(\mathcal{A}, W) \not\models KF$[2].
$(\mathcal{A}, W) =\!| MF$ iff $(\mathcal{A}, W) \not\models \neg MF$.
$(\mathcal{A}, W) =\!| F \wedge G$ iff $(\mathcal{A}, W) =\!| F$ or $(\mathcal{A}, W) =\!| G$.
$(\mathcal{A}, W) =\!| F; G$ iff $(\mathcal{A}, W) =\!| F$ and $(\mathcal{A}, W) =\!| G$.

It is clear that if a formula G is of the form KF, $\neg KF$, MF or $\neg MF$ where F is a propositional literal, then its truth value in $(\mathcal{A}, W)$ will not depend on W and we call G a *subjective literal*. On the other hand, if G does not contain K or M, then its truth value in $(\mathcal{A}, W)$ will only depend on W and we call G an *objective formula*. In the case that G is subjective, we write $\mathcal{A} \models G$ instead of $(\mathcal{A}, W) \models G$, and $W \models G$ instead of $(\mathcal{A}, W) \models G$ in the case that G is objective. From the above semantic definition, we should note that subjective literals $\neg KL$ and $M\neg L$ (here L is a literal) are not semantically equivalent. This is because even if there is some $W \in \mathcal{A}$ such that $L \notin W$, i.e. $\mathcal{A} \models \neg KL$, we cannot conclude that there is a $W' \in \mathcal{A}$ satisfying $\neg L \in W'$[3].

An *epistemic logic program* is a finite set of rules of the form:

$$F \leftarrow G_1, \cdots, G_m, not\ G_{m+1}, \cdots, not\ G_n. \qquad (1)$$

In (1) F is of the form $F_1; \cdots; F_k$ and $F_1, \cdots, F_k$ are objective literals, $G_1, \cdots, G_m$ are objective or subjective literals, and $G_{m+1}, \cdots, G_n$ are objective literals. For an epistemic logic program Π, 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 W of ground literals is called a *belief set* of Π iff W is a minimal set of satisfying conditions: (i) for each rule $F \leftarrow G_1, \cdots, G_m$ from Π such that $W \models G_1 \wedge \cdots \wedge G_m$ we have $W \models F$; and (ii) if W contains a pair of complementary literals then we write $W = Lit$, here Lit denotes an inconsistent belief set.
Step 2. Let Π be an epistemic logic program not containing modal operators K and M and W be a set of ground literals in the language of Π. By Π_W we denote the result of (i) removing from Π all the rules containing formulas of the form *not* G such that $W \models G$ and (ii) removing from the rules in Π all other occurrences of formulas of the form *notG*. We call W a *belief set* of Π if it is belief set of Π_W.
Step 3. Finally, let Π be an arbitrary epistemic logic program and $\mathcal{A}$ a collection of sets of ground literals in its language. By $\Pi_{\mathcal{A}}$ we denote the epistemic logic program

[1]This example was due to Gelfond (Gelfond 1994)

[2]We use $(\mathcal{A}, W) \not\models \varphi$ to denote that $(\mathcal{A}, W) \models \varphi$ does not hold.

[3]If L is $\neg F$ where F is an atom, then we write $\neg L = \neg\neg F = F$

obtained from Π by (i) removing from Π all rules containing formulas of the form G such that G is subjective and $\mathcal{A} \not\models G$, and (ii) removing from rules in Π all other occurrences of subjective formulas.

Now we define that a collection $\mathcal{A}$ of sets of ground literals is a *world view* of Π if $\mathcal{A}$ is the collection of all belief sets of $\Pi_{\mathcal{A}}$. Consider the program Π about the eligibility of scholarship discussed at the beginning of this section, if we replace rule r_4 with the following rule:

$$r_4'\text{: } interview(x) \leftarrow \neg K eligible(x), \neg K \neg eligible(x),$$

then the epistemic logic program that consists of rules r_1, r_2, r_3, r_4', and r_5 will have a unique world views $\{\{highGPA(mike), eligible(mike), interview(mike)\}, \{fairGPA(mike), interview(mike)\}\}$, which will result in a *yes* answer to the query $interview(mike)$.

Main Complexity Results

Let Π be an epistemic logic program, and $\mathcal{L}_\Pi$ the set of all literals of the underlying program Π. We specify $KM(\mathcal{L}_\Pi) = \{KL, ML \mid L \in \mathcal{L}_\Pi\}$. Clearly $|KM(\mathcal{L}_\Pi)| = 2|\mathcal{L}_\Pi|$. Now we define an *epistemic valuation* $\mathcal{E}$ over $KM(\mathcal{L}_\Pi)$ as follows:

$$\mathcal{E} : KM(\mathcal{L}_\Pi) \longrightarrow \{\mathtt{T}, \mathtt{F}\}.$$

We also specify $\mathcal{E}(\neg KL) = \neg\mathcal{E}(KL)$ and $\mathcal{E}(\neg ML) = \neg\mathcal{E}(ML)$.

By applying an epistemic valuation to an epistemic logic program Π, we may eventually remove all subjective literals from Π. Formally, we define the *epistemic reduction* of Π under $\mathcal{E}$, denoted as $eReduct(\Pi, \mathcal{E})$, to be a disjunctive extended logic program obtained from Π by (1) removing all rules where G is a subjective literal occurring in the rules and $\mathcal{E}(G) = \mathtt{F}$, and (2) removing all other occurrences of subjective literals in the remaining rules (i.e. replacing those G with $\mathtt{T}$ due to $\mathcal{E}(G) = \mathtt{T}$). Now we can define the concept of epistemic coherence which plays an important role in our following investigation.

Definition 1 *(***Epistemic coherence***) Let $\mathcal{E}$ be an epistemic valuation and $eReduct(\Pi, \mathcal{E})$ the program defined as above. We say that $\mathcal{E}$ is* epistemically coherent *with Π (or say that $\mathcal{E}$ is Π's an epistemically coherent valuation) if the following conditions hold:*

1. *for each rule in Π that has been removed from $eReduct(\Pi, \mathcal{E})$, there is some subjective literal G occurring in this rule satisfying $\mathcal{A} \not\models G$, where $\mathcal{A}$ is the collection of all answer sets of $eReduct(\Pi, \mathcal{E})$; and*
2. *for each rule in $eReduct(\Pi, \mathcal{E})$ where its each subjective literal G has been removed, $\mathcal{A} \models G$.*

Example 1 Consider an epistemic logic program Π consists of the following rules:

$a; b \leftarrow,$
$c \leftarrow \neg Ka,$
$d \leftarrow \neg Kb.$

If we define an epistemic valuation over $KM(\mathcal{L}_\Pi)$ as

$$\mathcal{E} = \{\neg Ka, \neg Kb, Kc, Kd, K\neg a, \neg K\neg b, K\neg c, K\neg d, Ma, M\neg a, \neg Mb, M\neg b, Mc, M\neg c, \neg Md, M\neg d\}^4,$$

then $eReduct(\Pi, \mathcal{E})$ is the following program:

$a; b \leftarrow,$
$c \leftarrow,$
$d \leftarrow,$

which has two answer sets $\{a, c, d\}$ and $\{b, c, d\}$. In fact we can verify that $\mathcal{E}$ is Π's epistemically coherent valuation. $\square$

We should indicate that an epistemic valuation may contain conflict truth values on some subjective literals. For instance, the above $\mathcal{E}$ contains both Kc and $M\neg c$ which are not satisfiable together. However, as long as these conflict subjective literals do not occur in the program at the same time (e.g. Kc and $M\neg c$ are not in Π), the epistemic valuation can still be coherent with respect to this program. Theorem 3 in section 4 actually considers this case.

The following lemma establishes an important relationship between the world view semantics and the epistemic valuation.

Lemma 1 *Let Π be an epistemic logic program. Π has a world view if and only if Π has an epistemically coherent valuation. Moreover, if $\mathcal{E}$ is an epistemically coherent valuation of Π, then the collection of all answer sets of program $eReduct(\Pi, \mathcal{E})$ is a world view of Π.*

Proof: ($\Rightarrow$) Suppose $\mathcal{A}$ is a world view of Π. Then we can easily define an epistemic valuation based on $\mathcal{A}$. For each $\alpha L \in KM(\mathcal{L}_\Pi)$ (α is a modality K or M), we specify $\mathcal{E}(\alpha L) = \mathtt{T}$ iff $\mathcal{A} \models \alpha L$. Then according to Definition 1, it is easy to show that $\mathcal{E}$ is epistemically coherent with Π. Furthermore, we have $eReduct(\Pi, \mathcal{E}) = \Pi_{\mathcal{A}}$. So the collection of all $eReduct(\Pi, \mathcal{E})$'s answer sets is exactly $\mathcal{A}$.
($\Leftarrow$) Now suppose that there is an epistemic valuation $\mathcal{E}$ that is coherent with Π. Let $\mathcal{A}$ be the collection of $eReduct(\Pi, \mathcal{E})$'s all answer sets (note that $eReduct(\Pi, \mathcal{E})$ must have an answer set). Then we do transformation $\Pi_{\mathcal{A}}$ (Step 3 in the world view definition presented in section 2). From Definition 1, quite obviously, we have $\Pi_{\mathcal{A}} = eReduct(\Pi, \mathcal{E})$. This proves our result. $\square$

The key feature of Lemma 1 is: it will allow us to design a deterministic algorithm to check the existence of the world view for an epistemic logic program through the testing of the coherence of an epistemic valuation. The algorithm takes an exponential amount of time to terminate, but as we will show, it only uses space efficiently. Towards this end, we will describe a function called *Coherence* which takes an epistemic logic program Π and an epistemic valuation $\mathcal{E}$ as input, and returns **True** if $\mathcal{E}$ is epistemically coherent with Π, otherwise it returns **False**.

Before we can formally describe function *Coherence*, we need some technical preparations. Consider the set $\mathcal{L}_\Pi$ of all literals of program Π. If we list all literals in $\mathcal{L}_\Pi$ with certain order (note that $\mathcal{L}_\Pi$ is finite), then we can encode each subset (i.e. belief set) of $\mathcal{L}_\Pi$ as a binary string of length

[4] For simplicity, we use this notion to represent $\mathcal{E}(Ka) = \mathtt{F}$, $\mathcal{E}(Kb) = \mathtt{F}$, etc.. In this way, we can simply view $\mathcal{E}$ as a set.

$|\mathcal{L}_\Pi|$. For instance, if $\mathcal{L}_\Pi = \{L_1, L_2, L_3, L_4\}$ with the order $L_1L_2L_3L_4$, then a binary string 0011 represents the belief set $\{L_3, L_4\}$. Therefore, we can enumerate all belief sets of $\mathcal{L}_\Pi$ according to the order by generating the corresponding binary strings from $000\cdots0$ to $111\cdots1$ one by one. We use $BiString(\mathcal{L}_\Pi, i)$ to denote the ith binary string generated as above according to a certain order. In the above example, we may have $BiString(\{L_1, L_2, L_3, L_4\}, 3) = 0011$ which represents the belief set $\{L_3, L_4\}$.

In a similar way, we also encode an epistemic valuation as a binary string. In particular, each epistemic valuation $\mathcal{E}$ over $KM(\mathcal{L}_\Pi)$ is represented as a binary string of length $2|\mathcal{L}_\Pi|$. Again, if we give an order on set $KM(\mathcal{L}_\Pi)$, then all epistemic valuations over $KM(\mathcal{L}_\Pi)$ can be generated as binary strings one at a time. We use notion $BiString(KM(\mathcal{L}_\Pi), i)$ to denote the ith generated binary string in this way.

Example 2 Suppose $KM(\mathcal{L}_\Pi) = \{Ka, K\neg a, Kb, K\neg b, Ma, M\neg a, Mb, M\neg b\}$. Then we have $BiString(KM(\mathcal{L}_\Pi), 10) = 00001001$, which represents an epistemic valuation: $\mathcal{E} = \{\neg Ka, \neg K\neg a, \neg Kb, \neg K\neg b, Ma, \neg M\neg a, \neg Mb, M\neg b\}$. □

Encoding belief sets and epistemic valuations as binary strings is an important step in our following proof because this will make it possible to automatically generate all candidate belief sets and epistemic valuations for testing one by one while we only use polynomial space.

We also need a sub-function named *Testing* to test the coherence of an answer set of program $eReduct(\Pi, \mathcal{E})$ with respect to $\mathcal{E}$. Recall that program $eReduct(\Pi, \mathcal{E})$ is obtained from Π by applying epistemic valuation $\mathcal{E}$ to subjective literals in all rules of Π. Once we generate an answer set of $eReduct(\Pi, \mathcal{E})$, we need to check if this answer set can be part of a world view of the original Π (see Lemma 1).

Function *Testing* manipulates a collection of lists of subjective literals, denoted as $SubList(\Pi)$, for program Π. For each rule r in Π, there is a list in $SubList(\Pi)$ of form $(G_1, \epsilon_1)\cdots(G_k, \epsilon_k)$ where $G_1, \cdots, G_k$ are all subjective literals occurring in rule r, and each ϵ_i is a boolean variable to record the coherent status of G_i under the current answer set to against the given epistemic valuation $\mathcal{E}$. ϵ_i changes its value during the testing process. For each run, *Testing* checks one answer set of $eReduct(\Pi, \mathcal{E})$. *Testing* returns 1 if this answer set is coherent with the underlying epistemic valuation $\mathcal{E}$. Otherwise, *Testing* returns 0 and the checking process stops. If all answer sets of $eReduct(\Pi, \mathcal{E})$ are coherent with $\mathcal{E}$, then *Testing* will set each ϵ_i's value to be 1 when it terminates.

```
*function $Testing(S, \Pi, \mathcal{E})$ returns $\delta \in \{0, 1\}$*
input: $S$, $\Pi$, and $\mathcal{E}$;
output: $\delta \in \{0, 1\}$;
$SubList(\Pi) = \{List_r : (G_1, \epsilon_1)\cdots(G_k, \epsilon_k) \mid G_1, \cdots G_k$
                    occur in $r\}$;
Compute $eReduct(\Pi, \mathcal{E})$;
Initialize $SubList(\Pi)$: $\forall List_r : (G_1, \epsilon_1)\cdots(G_k, \epsilon_k) \in$
$SubList(\Pi)$, $\epsilon_i = 0$ if rule $r$ is retained in $eReduct(\Pi, \mathcal{E})$,
        otherwise $\epsilon_i = 1$;
01 begin
02   for each $List_r \in SubList(\Pi)$ do
03       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (KL, 0)$,
04           then $\epsilon_i = 1$ if $i = 1$ and $L \in S$,
06       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (KL, 1)$,
07           then $\epsilon_i = 0$ if $L \notin S$; return 0;
08       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg KL, 0)$,
09           then $\epsilon_i = 1$ if $L \notin S$;
10       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (ML, 0)$,
11           then $\epsilon_i = 1$ if $L \in S$;
12       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg ML, 0)$,
13           then $\epsilon_i = 1$ if $i = 1$ and $L \notin S$;
14       if $r$ is retained in $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg ML, 1)$,
15           then $\epsilon_i = 0$ if $L \in S$; return 0;
16       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (KL, 1)$,
17           then $\epsilon_i = 0$ if $i = 1$ and $L \in S$;
18       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (KL, 0)$;
19           then $\epsilon_i = 1$ if $L \notin S$;
20       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg KL, 1)$;
21           then $\epsilon_i = 0$ if $L \notin S$; return 0;
22       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (ML, 1)$;
23           then $\epsilon_i = 0$ if $L \in S$; return 0;
24       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg ML, 1)$;
25           then $\epsilon_i = 0$ if $L \notin S$;
26       if $r$ is removed from $eReduct(\Pi, \mathcal{E})$ and
                 $(G_i, \epsilon_i) = (\neg ML, 0)$;
27           then $\epsilon_i = 1$ if $L \in S$;
28   return 1;
28 end
```

It is easy to see that *Testing* runs in polynomial time (and uses polynomial space of course). Although *Testing* looks a bit tedious, its checking procedure is quite simple. The following example illustrates the detail of how *Testing* works.

Example 3 We consider program Π:

$r_1 : a; b \leftarrow,$
$r_2 : c \leftarrow \neg Md,$
$r_3 : d \leftarrow \neg Mc,$
$r_4 : e \leftarrow Kc,$
$r_5 : f \leftarrow Kd.$

For program Π, we define a structure named $SubList(\Pi)$ which consists of the following collection of lists - each corresponds to a rule in Π (in this case, each list only has one element):

$List_{r_1} : \emptyset,$
$List_{r_2} : (\neg Md, \epsilon),$
$List_{r_3} : (\neg Mc, \epsilon),$

$List_{r_4} : (Kc, \epsilon)$,
$List_{r_5} : (Kd, \epsilon)$.

Now we specify an epistemic valuation as follows:

$$\mathcal{E} = \{Ka, \neg K\neg a, Kb, \neg K\neg b, Kc, \neg K\neg c, \neg Kd, \neg K\neg d, \neg Ke, \neg K\neg e, \neg Kf, \neg K\neg f, Ma, M\neg a, Mb, M\neg b, Mc, M\neg c, \neg Md, \neg M\neg d, Me, M\neg e, Mf, M\neg f\}.$$

Then we have $eReduct(\Pi, \mathcal{E})$:

$r_1 : a; b \leftarrow$,
$r'_2 : c \leftarrow$,
$r'_4 : e \leftarrow$,

which has two answer sets $\{a, c, e\}$ and $\{b, c, e\}$.

Having $eReduct(\Pi, \mathcal{E})$, we initialize the values of those ϵ as follows: (1) for those retained rules in $eReduct(\Pi, \mathcal{E})$, we set $\epsilon = 0$; and (2) for those removed rules from $eReduct(\Pi, \mathcal{E})$, we set $\epsilon = 1$. So the initial state of $SubList(\Pi)$ is as follows:

$List_{r_1} : \emptyset$,
$List_{r_2} : (\neg Md, 0)$,
$List_{r_3} : (\neg Mc, 1)$,
$List_{r_4} : (Kc, 0)$,
$List_{r_5} : (Kd, 1)$.

To check whether these two answer sets are in some world view of Π, we check the coherence between these answer sets and $\mathcal{E}$ by manipulating structure $SubList(\Pi)$. First, we consider answer set $S_1 = \{a, c, e\}$. Since $\neg Md$ occurs in Π and $d \notin S_1$, it means the subjective literal $\neg Md$ in rule r_1 is coherent with the answer set S_1 under $\mathcal{E}$. In this case, we change the value of the corresponding ϵ in (Md, ϵ) to 1, i.e. $List_{r_2} : (Md, 1)$. On the other hand, since rule r_3 is removed from $eReduct(\Pi, \mathcal{E})$ and $c \in S_1$, it means that S_1 is already coherent with $\neg Mc$. So we do not make any change on ϵ in $List_{r_3} : (\neg Mc, \epsilon)$ where $\epsilon = 1$. Similarly, we have $List_{r_4} : (Kc, 1)$ and $List_{r_5} : (Kd, 1)$ after the first run of *Testing*. In the second run, answer set $S_2 = \{b, c, e\}$ is used to check the coherence. We can verify that there will have no any further change on ϵ values. Therefore, the collection of S_1 and S_2 is a world view of the original program Π. $\Box$

Now we are ready to give the formal description of function *Coherence* which plays a key role in the upper bound proof.

```
*function $Coherence(\Pi, \mathcal{E})$ returns Boolean*
input: $\Pi$ and $\mathcal{E}$;
output: Boolean;
01  begin
02    Compute $eReduct(\Pi, \mathcal{E})$;
03    $i = 0$;
04    while $i < 2^{|\mathcal{L}_\Pi|}$ do
05      Generate a $S = BiString(\mathcal{L}_\Pi, i)$;
06      $i = i + 1$;
07      Testing whether $S$ is an answer set of
            $eReduct(\Pi, \mathcal{E})$;
08      if $S$ is an answer set of $eReduct(\Pi, \mathcal{E})$
09        then $Flag = Testing(S, \mathcal{E}, \Pi)$;
10        if $Flag = 0$ then return False;
12    if $\forall(G, \epsilon) \in SubList(\Pi)$, $\epsilon = 1$ then return True
13    else return False;
14    end
```

Theorem 1 *Given an arbitrary epistemic logic program* Π*. Deciding whether* Π *has a world view is in PSPACE.*

Proof: To prove this result, we design an algorithm called *Consistency* that takes an epistemic logic program Π as input and returns **True** if Π has a world view and **False** otherwise. Then we will show *Consistency* runs in exponential time but only uses polynomial space.

```
*algorithm $Consistency(\Pi)$ returns Boolean*
input: $\Pi$;
output: Boolean;
01  begin
02    $i = 0$;
03    while $i < 2^{2|\mathcal{L}_\Pi|}$ do
04      Generate a $\mathcal{E} = BiString(KM(\mathcal{L}_\Pi), i)$;
05      $i = i + 1$;
06      $V = Coherence(\Pi, \mathcal{E})$;
07      if $V = \mathbf{True}$
08      then return True;
09    return False;
10    end
```

Now we prove two results of $Consistency(\Pi)$: (1) Π has a world view if and only if $Consistency(\Pi)$ returns **True** (note that $Consistency(\Pi)$ always terminates), and (2) $Consistency(\Pi)$ runs in exponential time but only uses polynomial space. Result (1) can be proved directly from Lemma 1. Here we give the proof of Result (2).

We first consider function *Coherence*. It is easy to see that line 02 computing $eReduct(\Pi, \mathcal{E})$ only takes linear time, and $eReduct(\Pi, \mathcal{E})$ only needs a polynomial space. Within **while** loop, line 05 to line 10 take polynomial time noting that S is a binary string of length $|\mathcal{L}_\Pi|$, and there are at most $2^{|\mathcal{L}_\Pi|}$ loops. So the time complexity for **while** loop is at most $\mathcal{O}(2^{|\mathcal{L}_\Pi|})$ and the whole execution of **while** loop only uses polynomial space.

Now let us at look algorithm *Consistency*. Within the **while** loop from line 04 to line 08, each time $\mathcal{E}$ is generated as a binary string of length $2|\mathcal{L}_\Pi|$, and the call of function *Coherence* takes at most $\mathcal{O}(2^{|\mathcal{L}_\Pi|})$ time with polynomial space. As there are at most $2^{2|\mathcal{L}_\Pi|}$ loops, the time complexity of *Consistency* is at most $\mathcal{O}(2^{3|\mathcal{L}_\Pi|})$, but the space use remains in polynomial. $\Box$

The following result shows that the PSPACE upper bound for the consistency problem of epistemic logic programs is also tight.

Theorem 2 *Given an arbitrary epistemic logic program* Π*. Deciding whether* Π *has a world view is PSPACE-hard.*

Proof: We take a quantified boolean formula (QBF) of the form $A = \exists a_1 \forall a_2 \cdots Q_n a_n A'$, where $Q_n = \forall$ if n is an even number, otherwise $Q_n = \exists$, and A' is in CNF whose all propositional variables are among $\{a_1, \cdots, a_n\}$. It is well known that deciding whether A is valid is PSPACE-complete (Papadimitriou 1995). Based on this result, we construct a nontrivial epistemic logic program Π_A from the given formula A and show that A is valid if and only if Π_A has a world view. Since the construction of Π_A can be done in polynomial time, this proves the result.

Our approach is inspired from the PSPACE lower bound proof for modal logic K (Blackburn, de Rijke, & Venema 2001). The intuitive idea is described as follows: for any QBF Φ, we can build so called *quantifier trees*, each of such trees captures a collection of assignments that make Φ true. As an example, let us consider formula $\exists p \forall q \exists r(\neg p \vee q) \wedge \neg r$. This formula is valid and has a unique quantifier tree as shown in Figure 1.

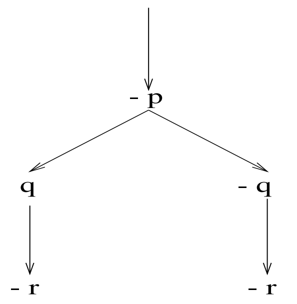

Figure 1: A quantifier tree.

The tree starts at level 0 and generates one branch to reach level 1 to represent p's one possible truth value (i.e. $\exists p$). Then the node at level 1 forces two branches to represent q's all possible truth values at level 2 (i.e. $\forall q$). Again, each node at this level generates one branch to represent one possible truth value of r at level 3 respectively (i.e. $\exists r$). Also note that all variables' truth values at a higher level are carried over to the lower levels (from root to leaves). Therefore, each leaf (i.e. node at level 3) represents an assignment on variables $\{p, q, r\}$ that evaluates formula $(\neg p \vee q) \wedge \neg r$ to be true, and the collection of all leaves forms one evaluation to make $\exists p \forall p \exists r(\neg p \vee q) \wedge \neg r$ true (in this example, there is only one such evaluation).

Based on this idea, we will construct a program Π_A to simulates the quantifier trees of formula A such that A is valid iff Π_A has a world view which corresponds to the collection of all leaves of a quantifier tree, and hence represents one possible evaluation to make A true. Furthermore, all world views of Π_A will represent all different evaluations that make A true.

Now we are ready to construct program Π_A. Let $V_A = \{a_1, \cdots, a_n\}$. we introduce new variables

$$\begin{aligned} V_{new} = \{&\overline{a_i} \mid \text{for } a_i \in V_A, i = 1, \cdots, n\} \cup \\ &\{a_1^j, \overline{a_1^j} \mid \text{for } a_1 \in V(A), j = 1, \cdots, n\} \cup \cdots \cup \\ &\{a_k^j, \overline{a_k^j} \mid \text{for } a_k \in V_A, \text{and } j = k, \cdots, n\} \cup \cdots \cup \\ &\{a_n^n, \overline{a_n^n} \mid \text{for } a_n \in V_A\} \cup \{neg, invalid\}. \end{aligned}$$

It is easy to see that $|V_{new}|$ is bounded by $\mathcal{O}(n^2)$. We explain the intuitive meaning of these newly introduced variable. Variable $\overline{a}_i$ is used to represent the negation of a_i. For each variable a_k that is universally quantified in A, i.e. k is an even number, we introduce variables a_k^j and $\overline{a_k^j}$ where $j = k, \cdots, n$ to represent a_k's values breaking into two branches at level k in the quantifier tree, and then carry over these values all the way down to the leaves. On the other hand, for each variable a_k that is existentially quantified in A, i.e. k is an odd number, we need to generate new quantifier trees in which variable a_k is assigned a different truth value. For this purpose, we also introduce variables a_k^j and $\overline{a_k^j}$ and $j = k, \cdots, n$ to represent a_k's two different possible values at different quantifier trees starting at level k, and then carry over these truth values down to the leaves respectively. We first specify groups of rules that simulate the quantifier tree.

Group G_1:

$a_1^1 \leftarrow \neg M \overline{a_1^1}$,
$\overline{a_1^1} \leftarrow \neg M a_1^1$,

$a_1^{i+1} \leftarrow a_1^i$,
$\overline{a_1^{i+1}} \leftarrow \overline{a_1^i}$, $i = 1, \cdots, (n-1)$,
$a_1 \leftarrow a_1^n$,
$\overline{a_1} \leftarrow \overline{a_1^n}$,

Group G_2:

$a_2^2; \overline{a_2^2} \leftarrow a_1^2$,
$a_2^2; \overline{a_2^2} \leftarrow \overline{a_1^2}$,

$a_2^{i+1} \leftarrow a_2^i$,
$\overline{a_2^{i+1}} \leftarrow \overline{a_2^i}$, $i = 2, \cdots, (n-1)$,
$a_2 \leftarrow a_2^n$,
$\overline{a_2} \leftarrow \overline{a_2^n}$,

$\cdots$,

Group G_k (k is an odd number):

$a_k^k \leftarrow a_{k-1}^k, \neg M \overline{a_k^k}$,
$\overline{a_k^k} \leftarrow a_{k-1}^k, \neg M a_k^k$,

$a_k^k \leftarrow \overline{a_{k-1}^k}, \neg M \overline{a_k^k}$,
$\overline{a_k^k} \leftarrow \overline{a_{k-1}^k}, \neg M a_k^k$,

$a_k^{i+1} \leftarrow a_k^i$,
$\overline{a_1^{i+1}} \leftarrow \overline{a_1^i}$, $i = k, \cdots, (n-1)$,
$a_k \leftarrow a_k^n$,
$\overline{a_k} \leftarrow \overline{a_1^n}$,

$a_k^{i+1} \leftarrow a_k^i$,
$\overline{a_k^{i+1}} \leftarrow \overline{a_k^i}$, $i = k, \cdots, (n-1)$,
$a_k \leftarrow a_k^n$,
$\overline{a_k} \leftarrow \overline{a_k^n}$,

Group G_{k+1}:

$a_{k+1}^{k+1}; \overline{a_{k+1}^{k+1}} \leftarrow a_k^{k+1}$,

$a_{k+1}^{k+1}; \overline{a_{k+1}^{k+1}} \leftarrow \overline{a_k^{k+1}}$,

$a_{k+1}^{i+1} \leftarrow a_{k+1}^{i}$,
$\overline{a_{k+1}^{i+1}} \leftarrow \overline{a_{k+1}^{i}}, i = (k+1), \cdots, (n-1)$,
$a_{k+1} \leftarrow a_{k+1}^{n}$,
$\overline{a_{k+1}} \leftarrow \overline{a_{k+1}^{n}}$,

Group G_n:
if n is an odd number, rules in this group are of the forms as in G_k, otherwise, they are of the forms as in G_{k+1}.

Let us take a closer look at rules specified above. Starting from the root, different truth values of variable a_1 are forced to be represented in different quantifier trees at level 1 due to the fact that a_1 is existentially quantified. Note that rules $a_1^1 \leftarrow \neg M\overline{a_1^1}$ and $\overline{a_1^1} \leftarrow \neg M a_1^1$ force a_1^1 and $\overline{a_1^1}$ can only be represented in different quantifier trees. Other rules in G_1 simply carry a_1^1 and $\overline{a_1^1}$ all the way down to the leaves (in different trees). On the other hand, since variable a_2 is universally quantified, the first two rules in G_2 force two branches in the quantifier tree where each branch represents one possible truth value of a_2. Similarly, other rules in G_2 are just to carry these truth values all the way down to the leaves. Other groups of rules have similar effects as G_1 and G_2 alternatively. Let $\Pi_A^{tree} = G_1 \cup \cdots \cup G_n$. It is easy to observe that $|\Pi_A^{tree}|$ is bounded by $\mathcal{O}(n^2)$.

Now we consider formula $A' = \{C_1, \cdots, C_m\}$. Suppose each $C \in A'$ is of the form $L_1 \vee \cdots \vee L_h$, where the corresponding propositional variables of $L_1, \cdots, L_h$ are among V_A. For each $C = L_1 \vee \cdots \vee L_h$, we specify a rule:

$r_c : neg \leftarrow \rho(\overline{L_1}), \cdots, \rho(\overline{L_h})$[5], where

$$\rho(\overline{L_i}) = \begin{cases} \overline{a} & \text{if } \overline{L_i} = \neg a \text{ where } a \in V_A \\ \overline{L_i} & \text{otherwise} \end{cases}$$

We denote the collection of all such rules as Π_A^{clause}. Finally we define a rule

$\Pi_A^{invalid}$:
$invalid \leftarrow neg, not\ invalid$.

We define $\Pi_A = \Pi_A^{tree} \cup \Pi_A^{clause} \cup \Pi_A^{invalid}$. We will show that A is valid iff Π_A has a world view. In fact, this is quite obvious. Since there is one-to-one correspondence between the world views of Π_A^{tree} and the collection of assignments under quantification $\exists a_1 \forall a_2 \cdots Q_n a_n$, we know that if A is valid, then in any case, variable neg will not be derived from Π_A, and therefore, the rule in $\Pi_A^{invalid}$ will never be triggered in Π_A. So program Π_A can be reduced to Π_A^{tree}, which has a world view. On the other hand, if Π_A has a world view, it concludes that all rules in Π_A^{clause} and the rule in $\Pi_A^{invalid}$ cannot be triggered. This implies that each world view of Π_A corresponds a collection of assignments on $\{a_1, \cdots, a_n\}$ which evaluates A to be true. $\Box$

[5] $\overline{L_i}$ stands for the negation of L_i.

Given an epistemic logic program Π and a (ground) literal L, we say that L is *entailed* from Π, if for Π's each world view $\mathcal{A}$ and $\forall W \in \mathcal{A}$, we have $L \in W$. From Theorems 1 and 2 and the fact that PSPACE=co-PSPACE (Blackburn, de Rijke, & Venema 2001), we have the following result.

Proposition 1 *Let* Π *be an epistemic logic program and* L *a literal. Deciding whether* $\Pi \models L$ *is PSPACE-complete.*

Computing World Views of Epistemic Logic Programs

With a slight modification on algorithm *Consistency*, we actually obtain an algorithm to compute world views of an epistemic logic program.

```
*algorithm $WView(\Pi)$ returns a world view or **Failed***
input: $\Pi$;
output: A world view of $\Pi$, or **Failed**;
01  begin
02    $i = 0$;
03    while $i < 2^{2|\mathcal{L}_\Pi|}$ do
04        Generate $\mathcal{E} = BiString(KM(\mathcal{L}_\Pi), i)$;
05        $i = i + 1$;
06        Compute $eReduct(\Pi, \mathcal{E})$;
07        Compute $\mathcal{A}$ = collection of all
              $eReduct(\Pi, \mathcal{E})$'s answer sets;
08        if $\mathcal{A} = \emptyset$ then return Failed;
09        Test whether $\mathcal{E}$ is epistemically coherent with $\Pi$;
10        if yes then return $\mathcal{A}$;
11    return Failed;
12    end
```

Algorithm *WView* can also be made to compute all world views of Π. Note that line 07 can be simply implemented by calling some answer set solver for disjunctive logic programs such as dlv (Leone, Rullo, & Scarcello 1996). We observe that in the worst case, the **while** statements from line 03 to line 10 contains $2^{2|\mathcal{L}_\Pi|}$ loops. So in general, *WView* is quite expensive.

But as we will show next, under some conditions, the efficiency of *WView* can be significantly improved. We first present a useful definition. If $\mathcal{E}$ is an epistemic valuation over $KM(\mathcal{L}_\Pi)$, we define $\mathcal{E}|_\Pi$ to be a *restriction* of $\mathcal{E}$ on Π, if $\mathcal{E}|_\Pi$ is a subset of $\mathcal{E}$ only containing truth values on those KL and ML occurring in Π.

Proposition 2 *Let* Π *be an epistemic logic program,* $\mathcal{E}$ *an epistemic valuation over* $KM(\mathcal{L}_\Pi)$*, and* $\mathcal{E}|_\Pi$ *a restriction of* $\mathcal{E}$ *on* Π*. Then* $\mathcal{E}$ *is epistemically coherent with* Π *if and only if* $\mathcal{E}|_\Pi$ *is epistemically coherent with* Π*.*

Proof: The result directly follows from Definition 1. $\Box$

With Proposition 2, *WView* can significantly improve its efficiency sometimes, because usually an epistemic logic program only contains a small number of subjective literals in its rules. Consider Example 1 presented in section 3, where we have $|\mathcal{E}| = 16$. However, since only Ka and Kb occur in Π, we have $\mathcal{E}|_\Pi = \{\neg Ka, \neg Kb\}$. Hence by applying Proposition 2, *WView* reduces its 2^{16} loops to only 4 loops in the worst case!

Now we consider other conditions under which *WView*'s efficiency can be improved as well. We can see that in *WView* another most expensive computation for generating a world view of Π is line `09` - testing the $\mathcal{E}$'s (or $\mathcal{E}|_{\Pi}$'s) epistemic coherence with Π[6]. This is because $\mathcal{A}$ (computed from line `07`) may contain exponential number of answer sets. Consequently testing the epistemic coherence may have to go through all these answer sets. So necessary optimization on this step is important.

Theorem 3 *Let Π be an epistemic logic program, $\mathcal{E}$ an epistemic valuation over $KM(\mathcal{L}_{\Pi})$. Then the following results hold:*

1. *Suppose that $KL, M\neg L \in \mathcal{E}$ and both occur in some rules in Π (maybe in different rules) as the only subjective literals (i.e. no $\neg$ sign in front). Then $\mathcal{E}$ is not epistemically coherent with Π or Π only has the inconsistent world view $\{Lit\}$;*
2. *Suppose that αL (α is K or M) occurs in some rules in Π as a subjective literal (i.e. no $\neg$ sign in front) and is in $\mathcal{E}$, and $\mathcal{E}'$ is another epistemic valuation that only differs with $\mathcal{E}$ on αL (i.e. $\neg\alpha L$ in $\mathcal{E}'$). If all rules containing αL also contains some subjective literal G where $\neg G \in \mathcal{E}$, then $\mathcal{E}$ is not epistemically coherent with Π iff $\mathcal{E}'$ is not epistemically coherent with Π.*

Proof: We first prove Result 1. Without loss of generality, we assume that there two rules are of the following forms in Π:

$r_1 : head(r_1) \leftarrow \cdots, KL, \cdots,$
$r_2 : head(r_2) \leftarrow \cdots, M\neg L, \cdots.$

Since KL and $M\neg L$ are the only subjective literal occurrences in r_1 and r_2 respectively, both rules are retained in $eReduct(\Pi, \mathcal{E})$ by removing KL and $M\neg L$ respectively. Then from Definition 1, we can see that $\mathcal{E}$ must be not epistemically coherent with Π since for the collection of all consistent answer sets of $eReduct(\Pi, \mathcal{E})$, $\mathcal{A} \models KL$ implies $\mathcal{A} \not\models M\neg L$ and *vice versa*. On the other hand, if $\mathcal{A} \not\models KL$ and $\mathcal{A} \not\models M\neg L$, it also concludes that $\mathcal{E}$ is not epistemically coherent with Π. The only case that $\mathcal{E}$ becomes epistemically coherent with Π is that Π has the inconsistent world view $\mathcal{A} = \{Lit\}$ so that $\mathcal{A} \models G$ for any subjective literals.

Now we prove Result 2. Suppose any rule containing αL in Π is of the form:

$r : head(r) \leftarrow \cdots, G, \alpha L, \cdots.$

Since $\alpha L \in \mathcal{E}$ and $\neg G \in \mathcal{E}$, it is easy to observe that $eReduct(\Pi, \mathcal{E}') = eReduct(\Pi, \mathcal{E})$, from which we know that the collection $\mathcal{A}$ of all answer sets of $eReduct(\Pi, \mathcal{E})$ is also the collection of all answer sets of $eReduct(\Pi, \mathcal{E}')$. Therefore, according to Definition 1, $\mathcal{E}$ is not epistemically coherent with Π iff $\mathcal{E}'$ is not epistemically coherent with Π. $\Box$

Theorem 3 provides two major conditions that can be used to simplify the process of epistemic coherence testing. Condition 1 simply says that any $\mathcal{E}$ epistemically coherent with Π cannot have both KL and $M\neg L$ to be true at the same time. Condition 2, on the other hand, presents a case that some αL's truth value does not have impact to the coherence of an epistemic valuation even if αL occurs in Π. This is particularly useful since $\mathcal{E}$ and $\mathcal{E}'$ are generated successively in *WView*. So if one valuation has been identified to be not epistemically coherent, there is no need to test the other one.

We should also mention that the efficiency for computing a world view of an epistemic logic program may also be improved if we revise the algorithm to be nondeterministic with proper optimization strategies for guessing epistemic valuations.

Computational Properties of Nested Epistemic Logic Programs

Wang and Zhang's nested epistemic logic programs (NELPs) (Wang & Zhang 2005) is a generalization of nested logic programs (Lifschitz, Tang, & Turner 1999) and Gelfond's epistemic logic programs under a unified language, in which nested expressions of formulas with knowledge and belief modal operators are allowed in both the head and body of rules in a logic program. Such generalization is important because it provides a logic programming framework to represent complex nonmonotonic epistemic reasoning that usually can only be represented in other nonmonotonic epistemic logics (Baral & Zhang 2005; Meyer & van der Hoek 1995).

Now we present key concepts and definitions of NELPs. Readers are referred to (Wang & Zhang 2005) for details. The semantics of NELPs, named *equilibrium views*, is defined on the basis of the epistemic HT-logic. Let $\mathcal{A}$ be a collection of sets of (ground) atoms[7] An *epistemic HT-interpretation* is defined as an ordered tuple $(\mathcal{A}, I^H, I^T)$ where I^H, I^T are sets of atoms with $I^H \subseteq I^T$. (H, T are called *tense* to represent *here* and *there* (Lifschitz, Pearce, & Valverde 2001)). If $I^H = I^T$, we say $(\mathcal{A}, I^H, I^T)$ is *total*. Also note that we do not force $I^H \in \mathcal{A}$ or $I^T \in \mathcal{A}$ (i.e. I^H, I^T may or may not be in $\mathcal{A}$). Then for $t \in \{H, T\}$, $(\mathcal{A}, I^H, I^T, t)$ *satisfies* a formula F, denoted as $(\mathcal{A}, I^H, I^T, t) \models F$, is defined as follows:

- for any atom F, $(\mathcal{A}, I^H, I^T, t) \models F$ if $F \in I^t$.
- $(\mathcal{A}, I^H, I^T, t) \not\models \bot$.
- $(\mathcal{A}, I^H, I^T, t) \models KF$ if $\forall J^H, J^T \in \mathcal{A}$ ($J^H \subseteq J^T$), $(\mathcal{A}, J^H, J^T, t) \models F$.
- $(\mathcal{A}, I^H, I^T, t) \models MF$ if $\exists J^H, J^T \in \mathcal{A}$ ($J^H \subseteq J^T$), $(\mathcal{A}, J^H, J^T, t) \models F$.
- $(\mathcal{A}, I^H, I^T, t) \models F \wedge G$ if $(\mathcal{A}, I^H, I^T, t) \models F$ and $(\mathcal{A}, I^H, I^T, t) \models G$.
- $(\mathcal{A}, I^H, I^T, t) \models F \vee G$ if $(\mathcal{A}, I^H, I^T, t) \models F$ or $(\mathcal{A}, I^H, I^T, t) \models G$.
- $(\mathcal{A}, I^H, I^T, t) \models \neg(\neg F \wedge \neg G)$.
- $(\mathcal{A}, I^H, I^T, t) \models F \rightarrow G$ if for every t' with $t \leq t'$,

[6]According to Proposition 2, it is easy to show that $eReduct(\Pi, \mathcal{E})$ is identical to $eReduct(\Pi, \mathcal{E}|_{\Pi})$.

[7]NELPs are first defined without considering strong negation, and then they are extended to the case of strong negation. For simplicity, here we do not consider such extension because this does not affect our results.

$(\mathcal{A}, I^H, I^T, t') \not\models F$ or $(\mathcal{A}, I^H, I^T, t') \models G$.
- $(\mathcal{A}, I^H, I^T, t) \models \neg F$ if $(\mathcal{A}, I^H, I^T, t) \models F \to \bot$.

Note that if F does not contain knowledge or belief operators, $(\mathcal{A}, I^H, I^T, t) \models F$ is irrelevant to $\mathcal{A}$, while if F is a subjective literal, $(\mathcal{A}, I^H, I^T, t) \models F$ is irrelevant to I^H and I^T. A *model* of an epistemic theory Γ is an epistemic HT-interpretation $(\mathcal{A}, I^H, I^T)$ by which every formula in Γ is satisfied. A total HT-interpretation $(\mathcal{A}, I, I)$ is an *epistemic equilibrium model* of Γ if it is a model of Γ and for each proper $J \subset I$, $(\mathcal{A}, J, I)$ is not a model of Γ. Now let Π be a nested epistemic logic program. We use $E(\Pi)$ to denote the epistemic theory obtained from Π by translating ";" to "$\vee$", "," to "$\wedge$" and "*not*" to "$\neg$". Then we define a collection $\mathcal{A}$ of sets of atoms to be an *equilibrium view* of Π if $\mathcal{A}$ is a maximal such collection satisfying $\mathcal{A} = \{I \mid (\mathcal{A}, I, I)$ is an epistemic equilibrium model of $E(\Pi)\}$. From these definitions, we have the following result.

Lemma 2 *A nested epistemic logic program Π has an equilibrium view if and only if the epistemic theory $E(\Pi)$ has an epistemic equilibrium model.*

Proof: Suppose Π has an equilibrium view $\mathcal{A}$. Then according to the definition, there exists at least one epistemic equilibrium model of $E(\Pi)$ that has the form $(\mathcal{A}, I, I)$. On other hand, if $E(\Pi)$ has an epistemic equilibrium model $(\mathcal{A}, I, I)$, then $\mathcal{A}$ will be an equilibrium view of Π if $\mathcal{A}$ satisfies the condition mentioned in the definition. Otherwise, there must be another epistemic equilibrium model $(\mathcal{A}', I', I')$ of $E(\Pi)$ such that $\mathcal{A}'$ is an equilibrium view of Π. $\Box$

Example 4 Consider the following nested epistemic logic program Π:

$Ka; Kb \leftarrow,$
$c \leftarrow Ka, not\ Mb,$
$d \leftarrow Kb, not\ Ma.$

From above definitions, Π has two equilibrium views $\{\{a, c\}\}$ and $\{\{b, d\}\}$. $\Box$

Theorem 4 *Given a nested epistemic logic program Π. Deciding whether Π has an equilibrium view is in PSPACE.*

Proof: According to Lemma 2, we only need to check whether $E(\Pi)$ has an epistemic equilibrium model. Given an epistemic HT-interpretation of the form $(\mathcal{A}, I, I)$, we know that the truth value of a formula not containing knowledge and belief operators will only depend on I, while the truth value of a subjective literal only depends on $\mathcal{A}$. So as in section 3, we can use the same way to *simulate* $\mathcal{A}$ by defining an epistemic valuation $\mathcal{E}$. However, since nested expressions are allowed in NELPs, this time, we define $\mathcal{E}$ not only over the set $KM(\mathcal{L}_\Pi)$, but also over the set $KM(\varphi_\Pi)$ of sub subjective formulas occurring in Π that are not in $KM(\mathcal{L}_\Pi)$ (e.g. $K(a \vee b)$). Note that $|KM(\varphi_\Pi)|$ is bound by the size of Π, i.e. the number of rules and the maximal length of these rules in Π. Let $KM(\Pi) = KM(\mathcal{L}_\Pi) \cup KM(\varphi_\Pi)$. Then $\mathcal{E}$ is defined over $KM(\Pi)$. In this way, a candidate equilibrium model of $E(\Pi)$ can be *simulated* as $(\mathcal{E}, I, I)$, where $I \subseteq \mathcal{L}_\Pi$.

As before, we encode $\mathcal{E}$ and I as binary strings so that both of them can be generated in some automatic way. According to the above definition, checking whether $(\mathcal{E}, I, I)$ is an equilibrium model of $E(\Pi)$ consists of two steps: (1) checking whether $(\mathcal{E}, I, I)$ is a model of $E(\Pi)$, which can be achieved in polynomial time; and (2) if yes, then checking whether for each $J \subset I$, $(\mathcal{E}, J, I)$ is not a model of $E(\Pi)$. Step (2) will take $\mathcal{O}(2^{|I|})$ amount of time, but each checking only uses polynomial space.

So by fixing an $\mathcal{E}$, for all possible $I \subset \mathcal{L}_\Pi$, checking whether $(\mathcal{E}, I, I)$ is an epistemic equilibrium models of $E(\Pi)$ will take $\mathcal{O}(2^{2^{|\mathcal{L}_\Pi|}})$ amount of time in the worst case, but only uses polynomial space. Since each $\mathcal{E}$ can be generated one by one, the whole process will terminate in time $\mathcal{O}(2^{2^{|\mathcal{L}_\Pi|}} \cdot 2^{|KM(\Pi)|})$ via only using polynomial space. $\Box$

It is interesting to note that the problem of consistency check for NELPs has the same upper bound as that of epistemic logic programs has, although the expressive power of NELPs has been significantly increased.

Theorem 5 *Given a nested epistemic logic program Π. Deciding whether Π has an equilibrium view is PSPACE-hard.*

Proof: Note that epistemic logic programs are a syntactic restriction of nested epistemic logic programs. Also in (Wang & Zhang 2005), it was proved that the equilibrium view semantics for NELPs coincides with the world view semantics for epistemic logic programs under this restriction. So the result directly follows from Theorem 2. $\Box$

Conclusion

We proved major complexity results for epistemic logic programs. The approach developed from our proof also yielded an algorithm for computing world views of epistemic logic programs, and can be used in proving the complexity results of nested epistemic logic programs. The results presented in this paper are important for us to understand the computational issues of epistemic reasoning under a logic programming setting.

Our work presented in this paper provides an answer to the initial computational problem of epistemic logic programs. Many related topics are worth our further study. We conclude this paper with an open problem: whether can we identify non-trivial subclasses of epistemic logic programs with lower complexity bounds?

References

Baral, C., and Zhang, Y. 2005. Knowledge updates: Semantics and complexity issues. *Artificial Intelligence* 164 (1-2):209–243.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*. Cambridge University Press.

Eiter, T., and Gottlob, G. 1995. On the complexity cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence* 15:289–323.

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

Halpern, J., and Moses, Y. 1992. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence* 54:311–379.

Leone, N.; Rullo, P.; and Scarcello, F. 1996. On the computation of disjunctive stable models. In *Proceedings of DEXA'96*, 654–666.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Transactions on Computational Logic* 7.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2(4):426–541.

Lifschitz, V.; Tang, L.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25:369–389.

Lobo, J.; Mendez, G.; and Taylor, S. 2001. Knowledge and the action description language $\dashv$. *Theory and Practice of Logic Programming* 1:129–184.

Meyer, J.-J. C., and van der Hoek, W. 1995. *Epistemic Logic for AI and Computer Science*. Cambridge University Press.

Papadimitriou, C. 1995. *Computational Complexity*. Addison Wesley.

Wang, K., and Zhang, Y. 2005. Nested epistemic logic programs. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR05)*, 279–290. Springer.

Watson, R. 2000. A splitting set theorem for epistemic specifications. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR-2000)*.

Zhang, Y. 2003. Minimal change and maximal coherence for epistemic logic program updates. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 112–117. Morgan Kaufmann Publishers, Inc.