A Modal Logic For Information System Security

Yun Bai[1] **Khaled M. Khan**[2]

[1] School of Computing and Mathematics
University of Western Sydney
Locked Bag 1797, Penrith South DC
NSW 1797, Australia
Email: ybai@scm.uws.edu.au

[2] Department of Computer Science and Engineering
Qatar University
Qatar
Email: k.khan@qu.edu.qa

Abstract

As a security mechanism, authorization or access control ensures that all accesses to the system resources occur exclusively according to the access polices and rules specified by the system security agent. Authorization specification has been extensively studied and a variety of approaches have been investigated. In this paper, we propose a knowledge oriented formal language to specify the system security policies and their reasoning in response to system resource access request. The semantics of our language is provided by translating our language into epistemic logic program in which knowledge related modal operators are employed to represent agents' knowledge in reasoning. We demonstrate how our authorization language handles the situation where the security agent's knowledge on access decision is incomplete.

Keywords: Access Control, Intelligent Systems, Formal Language, Authorization

1 Introduction

One important mechanism to protect information system is to control the access to the system resources. Authorization or access control is such a mechanism. It is to ensure that all accesses to the system resources occur exclusively according to the access polices and rules specified by the security agent of the information system. Authorizations or access control has been extensively studied in (Atluri et al. 2002), (Chomicki et al. 2000), (Fernandez et al. 1995), (Zhou et al. 2008) etc. and a variety of authorization specification approaches such as access matrix (Dacier et al. 1994), (Denning 1976), role-based access control (Crampton et al. 2008), access control in database systems (Bertino et al. 1996), authorization delegation (Murray et al. 2008), procedural and logical specifications (Bai et al. 2003), (Bertino et al. 2003) have been investigated. Since logic based approaches provide a powerful expressiveness (Fagin et al. 1995) as well as flexibility for capturing a variety of system security requirements, increasing work has been focusing on this aspect. Jajodia *et al* (Jajodia et al. 2001) proposed a logic language for expressing authorizations. They used predicates and rules to specify the authorizations; their work mainly emphasizes the representation and evaluation of authorizations. The work of Bertino *et al* (Bertino et al. 2000) describes an authorization mechanism based on a logic formalism. It mainly investigates the access control rules and their derivations. In their recent work (Bertino et al. 2003), a formal approach based on C-Datalog language is presented for reasoning about access control models.

Nevertheless, there were some limitations so far in these approaches. For instance, when the security agent does not have complete, specific information about the security domain, how to reason and answer access queries under such a scenario? For example, the agent currently does not know clearly who can access the classified file between Alice and Bob, but knows only one of them can. This can be specified by a disjunctive logic program (Baral 2003) as follows:

$AliceCanAccessFile \vee BobCanAccessFile \leftarrow,$
$AliceCanAccessFile \leftarrow not\ BobCanAccessFile,$
$BobCanAccessFile \leftarrow not\ AliceCanAccessFile.$

If a query asks if Alice can read the classified file, the agent will not be able to make the decision, because this program has two different answer sets: $\{AliceCanAccessFile\}$ and $\{BobCanAccessFile\}$. In fact, under many circumstances, using disjunctive logic programming to specify security policies is not sufficient to precisely handle incomplete information.

In this paper, we propose a knowledge based formal languages $\mathcal{L}^k$ to specify authorization domain with incomplete information in secure computer systems. We introduce modal logic to specify and reason about a security domain then translate the domain into an epistemic logic program. We show that our approach has an expressive power to describe a variety of complex security scenarios.

In our presentation, we assume the existence of a single, local system security officer or security agent administering the authorizations. This assumption enables us to concentrate on a single administering agent system and hence avoids the problem of coordination among multi agents. Access control in a multi-security system or administered by multi-security agents is out of the scope of this paper.

The rest of the paper is organized as follows. Sec-

This publication was made possible by a grant from the Qatar National Research Fund under its NPRP Grant No. 09-079-1-013. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the Qatar National Research Fund.

tion 2 describes language $\mathcal{L}^k$ by outlining its syntax and gives some authorization policy examples specified by the language. Section 3 explains the semantics of language $\mathcal{L}^k$. We start by introducing a general overview of epistemic logic program, then map the domain description specified by $\mathcal{L}^k$ into the logic program, and we give some examples to show the process and the domain description and its corresponding logic program. In section 4, we present a case study to demonstrate the reasoning of our system. Section 5 outlines the implementation issue. Section 6 concludes the paper with some remarks.

2 The syntax of a high level language $\mathcal{L}^k$

In this section we define the basic syntax of a high level language $\mathcal{L}^k$ which embeds a modal operator K to represent an agent's knowledge about access control policies.

Language $\mathcal{L}^k$ includes the following disjoint sorts for *subject* $(S, S_1, S_2, \cdots$ *for the constants and* $s, s_1, s_2, \cdots$ *for the variables)*, *group-subject* $(G, G_1, G_2, \cdots$ *for the constants and* $g, g_1, g_2, \cdots$ *for the variables)*, *access-right* $(A, A_1, A_2, \cdots$ *for the constants and* $a, a_1, a_2, \cdots$ *for the variables)*, *group-access-right* $(GA, GA_1, GA_2, \cdots$ *for the constants and* $ga, ga_1, ga_2, \cdots$ *for the variables)*, *object* $(O, O_1, O_2, \cdots$ *for the constants and* $o, o_1, o_2, \cdots$ *for the variables)*, *group-object* $(GO, GO_1, GO_2, \cdots$ *for the constants and* $go, go_1, go_2, \cdots$ *for the variables)* together with predicate symbols *holds* which takes arguments as *subject* or *group-subject, access-right* or *group-access-right* and *object or group-object* respectively; $\in$ which takes arguments as *subject* and *group-subject* or *access-right* and *group-access-right* or *object* and *group-object* respectively; $\subseteq$ whose both arguments are *group-subjects*, *group-access-rights* or *group-objects* and logic connectives $\wedge$ and $\neg$. We also introduce a modal operator K to represent what an agent *knows* to be true. We use $Khold(Sue, Read, File)$ to represent that "it's believed that Sue can read File".

In $\mathcal{L}^k$, a subject S has *read* right to an object $FILE$ is represented as $holds(S, read, FILE)$. It defines an atomic formula of the language. we define a *fact* f to be an atomic formula or its negation. A *ground fact* is a fact without variable occurrence. We view $\neg\neg f$ as f. A *fact expression* ϕ of $\mathcal{L}^k$ is defined as follows: (i) each fact ϕ is a fact expression; (ii) if ϕ and ψ are fact expressions, then $\phi \wedge \psi$ and $\phi \vee \psi$ are also fact expressions. A *ground fact expression* is a fact expression without variable occurrence. A ground fact expression is called a *ground instance* of a fact expression if this ground fact expression is obtained from the fact expression by replacing each of its variable occurrence with the same sort constant. A fact expression ϕ is called *conjunctive* (or *disjunctive*) if it is of the form $\phi_1 \wedge \cdots \wedge \phi_n$ (or $\phi_1 \vee \cdots \vee \phi_n$ respectively), where each φ_i is a fact.

Now we are ready to formally define the propositions in $\mathcal{L}^k$. An *initial proposition* in $\mathcal{L}^k$ is defined as

$$\textbf{initially } \phi \tag{1}$$

where ϕ is either a conjunctive or disjunctive fact expression. That is, ϕ is of the form $\phi_1 \wedge \cdots \wedge \phi_n$ or $\phi_1 \vee \cdots \vee \phi_n$, where each ϕ_i is a fact.

An *objective proposition* is an expression of the form

$$\phi \textbf{ if } \psi \textbf{ with absence } \gamma \tag{2}$$

where ϕ is either a conjunctive or disjunctive fact expression, ψ and γ are two conjunctive fact expressions.

A *subjective proposition* is an expression of the form

$$\phi \textbf{ if } \psi \textbf{ with absence } \gamma \textbf{ knowing } \beta, \tag{3}$$

or

$$\phi \textbf{ if } \psi \textbf{ with absence } \gamma \textbf{ not knowing } \beta, \tag{4}$$

where ϕ is a conjunctive or disjunctive fact expression, and ψ, γ and β are conjunctive fact expressions.

A proposition is called a *ground proposition* if it does not contain variables. A *policy domain description* D in $\mathcal{L}^k$ is a finite set of initial propositions, objective propositions and subjective propositions.

In the following, we describe a few complex security scenarios using language $\mathcal{L}^k$, and demonstrate that $\mathcal{L}^k$ is an expressive language to represent incomplete information, default information, and agents' knowledge in relation to various access control situations.

Example 1 *The example mentioned in the introduction can be represented by a domain description:*

> **initially** $holds(Alice, Access, File) \vee holds(Bob, Access, File)$,
> $holds(Alice, Access, File)$ **if** $\neg holds(Bob, Access, File)$,
> $holds(Bob, Access, File)$ **if** $\neg holds(Alice, Access, File)$

Here the initial fact $holds(Alice, Access, File) \vee holds(Bob, Access, File)$ *represents an incomplete information about Alice and Bob's access right to the file.*

Example 2 *Consider a domain description D consists of the following propositions:*

> **initially** $holds(S, Own, O)$,
> $holds(S, Write, O)$ **if** $holds(S, Own, O)$
> **with absence** $\neg holds(S, Write, O)$,
> $\neg holds(S, Own, O))$ **if** $\neg holds(S, Read, O)$.

This domain description expresses the following policies: initially subject S owns object O. If there is no evidence that S cannot write on O is absent from the domain, then S has write right on O, and S will no longer owns O if somehow S cannot read O anymore. Here **with absence** $\neg holds(S, Write, O)$ *represents a default information. As long as there is no clear information indicating* $\neg holds(S, Write, O)$*, it would be assumed that S can write O.*

Example 3 *Let us look at another example. A policy says that if a subject group G can read file F, then a member* S_1 *of G will be assumed to be able to read F as well if we don't know that* S_1 *cannot read F. This can be specified by the following propositions:*

> **initially** $holds(G, Read, F)$,
> **initially** $S_1 \in G$,
> $holds(S_1, Read, F)$ **if**
> $holds(G, Read, F)$, $S_1 \in G$,
> **not knowing** $\neg holds(S_1, Read, F)$

This example represents a policy involving agent (subject)'s knowledge for making decision. As we will show next, the semantics of knowledge in $\mathcal{L}^k$ *will be defined based on epistemic logic programming.*

3 Semantics of $\mathcal{L}^k$

Given a domain description D, we will translate it into an epistemic logic program $\Pi(D)$, then the semantics of D will be defined based on the *world view semantics* of program $\Pi(D)$.

In the following, we first introduce epistemic logic programs, and then define the semantics of $\mathcal{L}^k$.

3.1 Epistemic logic programs

In this section, we present a general overview on epistemic logic programs. Gelfond extended the syntax and semantics of disjunctive logic programs to allow the correct representation of incomplete information in the presence of multiple extensions (Gelfond 1994). 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". In this paper we will 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 the pair $(\mathcal{A}, W)$, where $\mathcal{A}$ is a collection of sets of ground literals which is also simply called is a collection of *belief sets*, and 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 is denoted by $(\mathcal{A}, W) =\!| F$. They are defined as follows.

$(\mathcal{A}, W) \models p$ iff $p \in W$ where p is a propositional 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 $(\mathcal{A}, W) \models G$.
$(\mathcal{A}, W) \models F\ or\ G$ 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 $(\mathcal{A}, W) \not\models KF$[1].
$(\mathcal{A}, W) =\!| MF$ iff $(\mathcal{A}, W) \not\models MF$.
$(\mathcal{A}, W) =\!| F \wedge G$ iff $(\mathcal{A}, W) =\!| F$ or $(\mathcal{A}, W) =\!| G$.
$(\mathcal{A}, W) =\!| F\ or\ G$ iff $(\mathcal{A}, W) =\!| F$ and $(\mathcal{A}, W) =\!| G$.

It is worth mentioning that since belief set W allows both positive and negative propositional atoms, in Gelfond's semantics, $(\mathcal{A}, W) =| \varphi$ is not equivalent to $(\mathcal{A}, W) \not\models \varphi$ in general. For instance, $(\{\{a,b\}\}, \{a,b\}) \not\models c$, but we do not have $(\{\{a,b\}\}, \{a,b\}) =| c$ (i.e. $(\{\{a,b\}\}, \{a,b\}) \models \neg c$). Consequently, here K and M are *not* dual modal operators here[2]. Consider $\mathcal{A} = \{\{a,b\}, \{a,b,\neg c\}\}$. Clearly we have $\mathcal{A} \models \neg K \neg c$. But having $\mathcal{A} \models Mc$ seems to be wrong.

If a formula G is of the form KF, $\neg KF$, MF or $\neg MF$ (where F is a propositional formula), then its truth value in $(\mathcal{A}, W)$ will not depend on W. In this case we call G a *subjective formula.* If F is a propositional literal, then we call KF, $\neg KF$, MF, and $\neg MF$ *subjective literals.* 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* or objective literal if G is a propositional literal. In the case that G is subjective, we simply 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. In general, we simply write $A \models G$ if for each $W \in A$, we have $(\mathcal{A}, W) \models G$. each $\mathcal{A}$

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

$$F \leftarrow G_1, \cdots, G_m, not\ G_{m+1}, \cdots, not\ G_n. \qquad (5)$$

In (5), $m, n \geq 0$, F is of the form $F_1\ or\ \cdots\ or\ F_k$ $(k \geq 1)$ 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 $\mathcal{P}$, 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 $W = Lit$, i.e. W is an inconsistent belief set[3].

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*.

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 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 $\Pi_{\mathcal{A}}$.

Example 4 *Consider a simple epistemic logic program* Π *consisting of the following rules:*

$a \vee b \leftarrow \neg Mc,$
$d \leftarrow Ka,$
$e \leftarrow b, not\ e.$

Let $\mathcal{A} = \{\{a,d\}\}$*, then from the above definition, we have its belief sets* Π_A*:*

$a \vee b \leftarrow,$
$d \leftarrow,$
$e \leftarrow b, not\ \neg e.$

Then it is easy to see that $\{a,d\}$ *is the only answer set of* $\Pi_{\mathcal{A}}$*. So* $\mathcal{A}$ *is a world view of* Π*. It can be also verify that* $\mathcal{A}$ *is the unique world view of* Π*.*

3.2 Translating a domain description into an epistemic logic program

Now we define the semantics of $\mathcal{L}^k$ based on the world view semantics of epistemic logic programs. Let D be

[1]We denote $(\mathcal{A}, W) \not\models \varphi$ iff $(\mathcal{A}, W) \models \varphi$ does not hold.

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

[3]Note that in our context, a belief set is simply a set of ground literals. Here a belief set of a program is a belief set that satisfies the conditions (i) and (ii).

a given domain description of $\mathcal{L}^k$, i.e. D is a finite set of propositions as illustrated in section 2.1. We specify an epistemic logic program $\Pi(D)$ translated from D as follows:

1. For an initial policy proposition (1): **initially** ϕ, if ϕ is a conjunctive fact expression $\phi_1 \wedge \cdots \wedge \phi_n$, then it is translated to a set of rules[4]:

   $$\phi_1 \leftarrow, \\ \cdots, \\ \phi_n \leftarrow,$$

   if ϕ is a disjunctive fact expression $\phi_1 \vee \cdots \vee \phi_n$, then it is translated to *one* rule:

   $$\phi_1 \vee \cdots \vee \phi_n \leftarrow,$$

2. For each objective access proposition (2): ϕ **if** ψ **with absence** γ, here $\psi = \psi_1 \wedge \cdots \wedge \psi_k$ and $\gamma = \gamma_1 \wedge \cdots \wedge \gamma_l$, if ϕ is a conjunctive fact expression $\phi_1 \wedge \cdots \wedge \phi_n$, then it is translated to a set of rules:

   $$\phi_1 \leftarrow \psi_1, \cdots, \psi_k, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \gamma_l, \\ \cdots, \\ \phi_n \leftarrow \psi_1, \cdots, \psi_k, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \gamma_l,$$

   if ϕ is a conjunctive fact expression $\phi_1 \vee \cdots \vee \phi_n$, then it is translated to *one* rule:

   $$\phi_1 \vee \cdots \vee \phi_n \leftarrow \psi_1, \cdots, \psi_k, \\ \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \gamma_l,$$

3. For each subjective access proposition (3): ϕ **if** ψ **with absence** γ **knowing** β, where $\psi = \psi_1 \wedge \cdots \wedge \psi_k$, $\gamma = \gamma_1 \wedge \cdots \wedge \gamma_l$, and $\beta = \beta_1 \wedge \cdots \wedge \beta_r$, if ϕ is a conjunctive fact expression $\phi_1 \wedge \cdots \wedge \phi_n$, then translate it to a set of rules:

   $$\phi_1 \leftarrow \psi_1, \cdots, \psi_k, \\ K\beta_1, \cdots, K\beta_r, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \beta_r, \\ \cdots, \\ \phi_n \leftarrow \psi_1, \cdots, \psi_k, \\ K\beta_1, \cdots, K\beta_r, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \beta_r,$$

   if ϕ is a disjunctive fact expression $\phi_1 \vee \cdots \vee \phi_n$, then translate it to *one* rule:

   $$\phi_1 \vee \cdots \vee \phi_n \leftarrow \psi_1, \cdots, \psi_k, \\ K\beta_1, \cdots, K\beta_r, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \beta_r,$$

4. For each subjective access proposition (4): ϕ **if** ψ **with absence** γ **not knowing** β, where $\psi = \psi_1 \wedge \cdots \wedge \psi_k$, $\gamma = \gamma_1 \wedge \cdots \wedge \gamma_l$, and $\beta = \beta_1 \wedge \cdots \wedge \beta_r$, if ϕ is a conjunctive fact expression $\phi_1 \wedge \cdots \wedge \phi_n$, then translate it to a set of rules:

   $$\phi_1 \leftarrow \psi_1, \cdots, \psi_k, \\ \neg K\beta_1, \cdots, \neg K\beta_r, \mathit{not}\ \gamma_1, \cdots, \\ \mathit{not}\ \beta_r, \\ \cdots, \\ \phi_n \leftarrow \psi_1, \cdots, \psi_k, \\ \neg K\beta_1, \cdots, \neg K\beta_r, \mathit{not}\ \gamma_1, \cdots, \\ \mathit{not}\ \beta_r,$$

   if ϕ is a disjunctive fact expression $\phi_1 \vee \cdots \vee \phi_n$, then translate it to *one* rule:

   $$\phi_1 \vee \cdots \vee \phi_n \leftarrow \psi_1, \cdots, \psi_k, \\ \neg K\beta_1, \cdots, \neg K\beta_r, \mathit{not}\ \gamma_1, \cdots, \mathit{not}\ \beta_r,$$

[4]Note that each φ_i is an atom or a negation of an atom.

Now we specify $\Pi(D)$ to be the collection of all rules translated from D by the above procedure. It is noted that $\Pi(D)$ is an epistemic logic program without modal operator M.

Since positions in D may contain variables, program $\Pi(D)$ may also contain variables. In this case, a ground epistemic logic program generated from $\Pi(D)$ by replacing each variable with all possible corresponding sort constants occurring in $\Pi(D)$. Without much confusion, we may still use notion $D(\Pi)$ to denote this corresponding ground program.

Definition 1 *Let D be a domain description of $\mathcal{L}^k$, $\Pi(D)$ the epistemic logic program translated from D as described above, and f a ground fact. We say that D* entails *f, denoted as $D \models f$, if $\Pi(D)$ has a world view, and for each world view $\mathcal{A}$ of $\Pi(D)$, $\mathcal{A} \models f$.*

Example 5 *Consider Example 3 presented in section 2. According to the above procedure, we can translate the domain description D as the following program $\Pi(D)$:*

$$\mathit{holds}(G, \mathit{Read}, F) \leftarrow, \\ S_1 \in G \leftarrow, \\ \mathit{holds}(S_1, \mathit{Read}, F) \leftarrow \mathit{holds}(G, \mathit{Read}, F), \\ S_1 \in G, \neg K \neg \mathit{holds}(S_1, \mathit{Read}, F).$$

Now suppose we need to answer a query whether S_1 can read file F, i.e. whether $D \models \mathit{holds}(S_1, \mathit{Read}, F)$. It is not difficult to see that program $\Pi(D)$ has a unique world view $\mathcal{A} = \{\{\mathit{holds}(G, \mathit{Read}, F), S_1 \in G, \mathit{holds}(S_1, \mathit{Read}, F)\}\}$, and $\mathcal{A} \models \mathit{holds}(S_1, \mathit{Read}, F)$. So we conclude that $D \models \mathit{holds}(S_1, \mathit{Read}, F)$.

4 A case study: Reasoning about knowledge in access control

In this section, we demonstrate a case study from which we show that our approach can overcome some difficulties in the reasoning about access control when incomplete information is involved.

We consider a typical hospital scenario that doctor assistants take responsibility to manage patients files and access relevant files and data from other department. In order to ensure the confidentiality of all patients' medical records, a number of authorization policies must be implemented in all departments in a hospital.

Suppose that Hobson is a heart specialist in a hospital. He is planning a by-pass surgery for his patient John, for that purpose, he needs to review John's all other recent medical records before the surgery. Alice and Sue are the personal assistants of Hobson. Each of them can access doctor Hobson's all patients' records, while Sue also takes responsibility to request patients' medical records from other departments in the hospital.

By using our language $\mathcal{L}^k$, we first formalize the general authorization policies across the hospital as follows:

$$\mathit{holds}(x, \mathit{Read}, \mathit{All\_heart\_records})\ \textbf{if knowing} \\ \mathit{assistant}(x, \mathit{Hobson}), \quad (6)$$

$$\mathit{holds}(x, \mathit{Read}, \mathit{y\_heart\_record})\ \textbf{if} \\ \mathit{holds}(x, \mathit{Read}, \mathit{All\_heart\_records}) \\ \wedge \mathit{patient}(y, \mathit{Hobson}), \quad (7)$$

$$\mathit{holds}(\mathit{Hobson}, \mathit{Read}, y)\ \textbf{if}\ \mathit{holds}(x, \mathit{Read}, y) \wedge \\ \mathit{assistant}(x, \mathit{Hobson}), \quad (8)$$

$$sendRequest(Sue, Read, y) \textbf{ if } request(Hobson, y) \textbf{ with absence } \neg sendRequest(Sue, Read, y), \qquad (9)$$

$$sendRequest(Alice, Read, y) \textbf{ if } request(Hobson, y) \textbf{ with absence } sendRequest(Sue, Read, y). (10)$$

$$waitingApproval(x, Read, y) \textbf{ if } sendRequest(x, Read, y) \wedge \textbf{ not knowing } approved(x, Read, y), \qquad (11)$$

$$approved(x, Read, y) \textbf{ if } sendRequest(x, Read, y) \wedge assistant(x, d) \wedge specialist(d), \qquad (12)$$

$$holds(x, Read, y) \textbf{ if } approved(x, Read, y). \qquad (13)$$

Let us take a closer look at these rules. Basically, rules (6) and (7) say that if it is known that x is a personal assistant of Doctor Hobson, then x can access (read) Doctor's all patients' heart records, and if someone is already permitted to read all patients' heart records, and y is a patient of Doctor, then this person can also read y's heart record. Note that rule (7) plays a role of inheritance for access control. Also, rule (8) implies the fact that once Doctor Hobson's assistant x obtains the access read for some patient record from other department, then Doctor Hobson should have the access right on this record obviously.

Rule (9) indicates that if Doctor Hobson has a request of accessing patient y's record from other departments, then *usually* Sue should send this request for approval. Note that this rule is defeasible due to **with absence**. For instance, if Sue is on leave, then $\neg sendRequest(Sue, Read, y)$ will be presented and hence this rule will not be initiated any more. Rule (10) describes the case that Alice will do Sue's duty when she is not available. On the other hand, rule (11) means that once a request is sent out, it is on the waiting status if no approval from that department is explicitly informed. Rule (12) states that the corresponding department will approve the request sent by x about y's record if x is a personal assistant of some doctor d who is a registered specialist of the hospital. Finally, rule (13) is quite straightforward that if x receives the approval of the department that holds patient record, x can then access y's record in that department.

Now suppose we have the following facts:

$$\textbf{initially } assistant(Alice, Hobson), \qquad (14)$$

$$\textbf{initially } assistant(Sue, Hobson), \qquad (15)$$

$$\textbf{initially } patient(John, Hobson), \qquad (16)$$

$$\textbf{initially } specialist(Hobson), \qquad (17)$$

$$\textbf{initially } request(Hobson, John\_generalHealth\_record), \qquad (18)$$

$$\textbf{initially } \neg sendRequest(Sue, Read, John\_generalHealth\_record), \qquad (19)$$

we would like to know how the access right "Read" for patient John's general health record can be obtained by Doctor Hobson. Let D be the domain description consisting of propositions (6) - (19). Then applying our translation procedure described in section 3.2, we can obtain the following epistemic logic program $\Pi(D)$:

$$\begin{array}{l}
holds(x, Read, All\_heart\_records) \leftarrow \\
\quad K assistant(x, Hobson), \\
holds(x, Read, y\_heart\_record) \leftarrow \\
\quad holds(x, Read, All\_heart\_records), \\
\quad patient(y, Hobson), \\
holds(Hobson, Read, y) \leftarrow \\
\quad holds(x, Read, y), assistant(x, Hobson), \\
sendRequest(Sue, Read, y) \leftarrow \\
\quad request(Hobson, y), \\
\quad not\ \neg sendRequest(Sue, Read, y), \\
sendRequest(Alice, Read, y) \leftarrow \\
\quad request(Hobson, y), \\
\quad not\ sendRequest(Sue, Read, y), \\
waitingApproval(x, Read, y) \leftarrow \\
\quad sendRequest(x, Read, y), \\
\quad \neg K approved(x, Read, y), \\
approved(x, Read, y) \leftarrow \\
\quad sendRequest(x, Read, y), assistant(x, d), \\
\quad specialist(d), \\
holds(x, Read, y) \leftarrow approved(x, Read, y), \\
assistant(Alice, Hobson) \leftarrow, \\
assistant(Sue, Hobson) \leftarrow, \\
patient(John, Hobson) \leftarrow, \\
specialist(Hobson, H) \leftarrow, \\
request(Hobson, John\_generalHealth\_record) \leftarrow, \\
\neg sendRequest(Sue, Read, \\
John\_generalHealth\_record) \leftarrow.
\end{array}$$

It is easy to see that $\Pi(D)$ has a unique world view A:
$\{\{assistant(Alice, Hobson), assistant(Sue, Hobson),$
$patient(John, Hobson), specialist(Hobson),$
$request(Hobson, John\_generalHealth\_record),$
$\neg sendRequest(Sue, Read, John\_generalHealth\_record)$
$sendRequest(Alice, Read, John\_generalHealth\_record),$
$holds(Alice, Read, All\_heart\_records)$
$holds(Sue, Read, All\_heart\_records),$
$holds(Alice, Read, John\_heart\_records)$
$holds(Sue, Read, John\_heart\_records),$
$approved(Alice, Read, John\_generalHealth\_record)$
$holds(Alice, Read, John\_generalHealth\_records)\}\},$

From A we can finally derive that the following results:

$$\begin{array}{l}
D \models sendRequest(Alice, Read, \\
\quad John\_generalHealth\_record), \\
D \models approved(Alice, Read, \\
\quad John\_generalHealth\_record), \\
D \models holds(Alice, Read, \\
\quad John\_generalHealth\_record), \\
D \models holds(Hobson, Read, \\
\quad John\_generalHealth\_record).
\end{array}$$

5 The implementation issues

A system for epistemic logic programming has been implemented. In this section we briefly outlined our implementation of our epistemic logic programming system and explain how our formal language $\mathcal{L}^k$ developed in this paper is fulfilled by the system.

The system we implemented is called World Views Solver, simply denoted as Wviews. The essential function of Wviews is to compute one or all world views (models) of an input epistemic logic program. To compute the world views of an epistemic logic program Π, Wviews first performs a reduction to transform Π into a traditional disjunctive logic program (DLP), then call dlv to compute the answer sets of $\Pi^{\mathcal{A}}$. The system structure is outlined as follows[5].

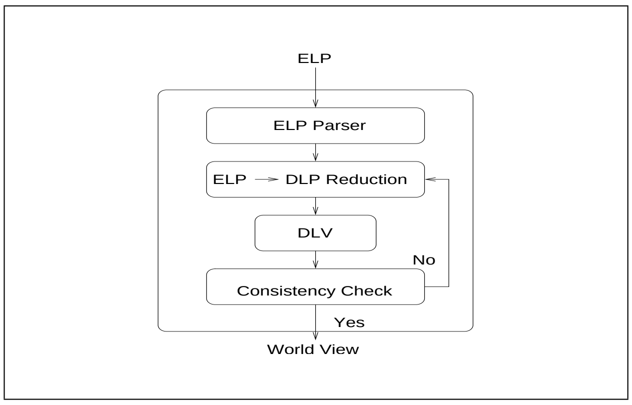

Figure 1: Wviews system structure.

As we mentioned in section 3, the semantics of $\mathcal{L}^k$ is defined in terms of the world view semantics of epsitemic logic programs. Having system Wviews, we can easily implement our policy language $\mathcal{L}^k$ in the following way: taking the domain description D as the input, which is a finite set of $\mathcal{L}^k$ propositions (see section 2), we implement a transformation procedure as illustrated in section 3, to translate D into an epistemic logic program $\Pi(D)$, then by calling system Wviews, we will be able to compute one or all world views (models) of $\Pi(D)$.

6 Conclusion

In this paper, we proposed a formal language $\mathcal{L}^k$ to specify security polices by an authorization domain with incomplete information. Different from previous policy specification languages, our formal language $\mathcal{L}^k$ has knowledge as its key feature to deal with incomplete domains. We specified the semantics of such knowledge oriented authorization specification language based on the well known the world view semantics of epistemic logic programs. The examples showed demonstrated that our approach has a rich expressive power to describe a variety of complex security requirements. Related semantic and computational properties of epistemic logic programs have been studied in (Zhang 2007), which will be help us to fully using the expressive power of epistemic logic programming to represent and reason about knowledge based authorization policies. This is our current research focus.

References

Atluri, V. & Gal, A. (2002), An authorization model for temporal and derived data: securing information protals, *ACM Transactions on Information and System Security*, Vol.5, No.1, pp. 62–94.

Bai, Y. & Varadharajan, V. (2003), On transformation of authorization policies, *Data and Knowledge Engineering*, Vol.45, No.3, pp. 333–357.

Baral, C. (2003), *Knowledge Representation, Reasoning, and Declarative Problem Solving*, MIT Press.

Bertino, E., Buccafurri, F., Ferrari, E. & Rullo, P. (2000), A Logic-based Approach for Enforcing Access Control. *Computer Security*, Vol.8, No.2-2, pp. 109–140.

Bertino, E., Catania, B., Ferrari, E. & Perlasca, P. (2003), A logical framework for reasoning about access control models, *ACM Transactions on Information and System Security*, Vol.6, No.1, pp. 71–127.

Bertino, E., Jajodia, S & Samarati, P (1996), Supporting multiple access control policies in database systems, *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 94–107.

Chomicki, J., Lobo, J. & Naqvi, S. (2000), A logical programming approach to conflict resolution in policy management, *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning*, pp. 121–132.

Crampton, J. & Khambhammettu, H. (2008), Delegation in role-based access control. *International Journal of Information Security*, Vol.7, pp. 123–136.

Dacier, M & Deswarte, Y. (1994), Privilege graph: an extension to the typed access matrix model, *Proceedings of European Symposium on Research in Computer Security*, pp. 319–334.

Denning, D.E. (1976), A lattice model of secure information flow, *Communication of ACM*, Vol.19, pp. 236–243.

Fagin, R., Halpern, J.Y., Moses, Y & Vardi, M.Y. (1995), *Reasoning about knowledge*. MIT Press.

Fernandez, E.B., France, R.B. & Wei, D (1995), A formal specification of an authorization model for object-oriented databases, *Database Security, IX: Status and Prospects*, pp. 95–109.

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

Jajodia, S., Samarati, P., Sapino, M.L. & Subrahmanian, V.S. (2001), Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, Vol.29, No.2, pp. 214–260, 2001.

Murray, T. & Grove, D. (2008), Non-delegatable authorities in capability systems. *Journal of Computer Security* Vol.16, pp. 743–759.

Zhang, Y. (2007), Epistemic reasoning in logic programs. In *in* 'Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)', pp. 647–652.

Zhou, J. & Alves-Foss, J. (2008), 'Security policy refinement and enforcement for the design of multilevel secure systems', *Journal of Computer Security*, Vol.16, pp. 107–131.

[5]The system details can be accessed from http://www.scm.uws.edu.au/ yan/Wviews.html.