A Logic for State Transformations in Authorization Policies

Yun Bai and Vijay Varadharajan
Distributed System and Network Security Research Unit
Dept. of Computing, University of W. Sydney, Australia
{ybai,vijay}@st.nepean.uws.edu.au

Abstract

In a multi-user, information-sharing system, authorization policy provides the ability to limit and control access to system, applications and information. In the real world, an authorization policy has temporal properties. That is, it needs to be updated to capture the changing requirements of applications, systems and users. These updates are implemented via transformation of authorization policies. In this paper, we propose a logic based approach to specify and to reason about state transformations in authorization policies. The authorization policy is specified using a policy base which comprises a finite set of facts and access constraints. We define the structure of the policy transformation and employ a model-based semantics to perform the transformation under the principle of minimal change. Furthermore, we extend model-based semantics by introducing preference ordering to resolve possible conflicts during transformation of policies. We also discuss the implementation of the model based transformation approach and outline the relevant algorithms.

1. Introduction

In a multi-user, information-sharing system, authorization policy provides the ability to limit and control access to system, applications and information, and to limit what entities can do with the information and the resources. In the real world, authorization policy has a temporal property, it needs to capture the *changing* requirements of applications, systems and users. This implies that situations can arise where some subjects (users or processes) can gain some access rights for some objects and at the same time can lose some access rights for the same or different objects. Representation, evaluation and analysis of such changes form an important part of the design of authorization policies. These changes are implemented via state transformation of authorization policies. In general such transformations can be *nonmonotonic* in that some users or subjects may *lose* certain rights. In this paper, we will discuss the design of authorization policies, their transformation, the mechanism for reasoning about nonmonotonic properties and their implementation.

In our system, authorization policy is specified by a policy base which is a finite set of facts and access constraints. The facts represent explicitly the access rights the subjects hold for the object. They are a Herbrand interpretation [11] of the language of our system. The access constraints, on the other hand, are rules which the authorization policy should satisfy. They are a set of closed first order formulas of the language.

We first define the structure of the policy transformation. The structure describes the preconditions that need to be satisfied before the transformation can be executed and the postconditions that become valid after the transformation. We employ *a model-based semantics* [11] in the computation of the transformation; with the model based approach, the transformation of a policy base is based on the Herbrand interpretation present in the policy base. The *principle of minimal change* [11] is used to guarantee that the change of the policy base after the transformation is as minimal as possible. We also introduce preference ordering in the model-based semantics to resolve possible conflicts. We discuss the implementation aspects of our system and analyze the complexity of the algorithms introduced.

Let us first briefly mention relevant related work in this area. The work by Woo and Lam [12] used a logic approach to represent and evaluate authorization policies; our work concentrates on the issue of representing and performing nonmonotonic transformation of authorization policies. The work by Sandhu and Suri addressed the issue of transformation of access rights [9, 10]. Their work was based on the access matrix, in which no constraint is presented explicitly. They used a procedure-based approach to represent the transformation. Our work uses a logic based approach and develops *a model based semantics* to formalize the transformation. This, as will be shown in this paper, is more powerful and feasible for implementation.

To the best of our knowledge, the work presented in this paper is the first one which uses a model-based semantics to formalize logic based transformation of authorization. We believe that our system provides a unified framework which can be used to model other methods. In fact, in the full version of this manuscript [1], we have shown that the scheme by Sandhu et al can be subsumed by our system.

The paper is organized as follows. Section 2 introduces the formal definition of the policy base and its use in the specification of authorization policies. Section 3 defines the transformation description, discusses the computation of the transformation and its nonmonotonic property. Section 4 extends the model-based semantics by introducing a preference ordering mechanism to resolve conflicts in transformations. Section 5 discusses the implementation issues. Finally, section 6 concludes the paper with some remarks.

2 A Formal Representation of Policy Base

In this section, we introduce a formal model for representing policy base based on a first order language. We give both syntactic and semantic descriptions of our policy base model.

2.1 The Language

Let $\mathcal{L}$ be a sorted first order language with equality, with four disjoint sorts for *subject, group-subject, access-right,* and *object* respectively. Assume $\mathcal{L}$ has the following vocabulary:

1. Sort *subject*: with subject constants $S, S_1, S_2, \cdots$, and subject variables $s, s_1, s_2, \cdots$.
2. Sort *group-subject*: with group subject constants $G, G_1, G_2, \cdots$, and group subject variables g, g_1, g_2.
3. Sort *access-right*: with access right constants $A, A_1, A_2, \cdots$, and access right variables $a, a_1, a_2, \cdots$.
4. Sort *object*: with object constants $O, O_1, O_2, \cdots$, and object variables $o, o_1, o_2, \cdots$.
5. A ternary predicate symbol *s-holds* which takes arguments as *subject, access-right* and *object* respectively.
6. A ternary predicate symbol *g-holds* which takes arguments as *group-subject, access-right* and *object* respectively.
7. A binary predicate symbol $\in$ which takes arguments as *subject* and *group-subject.*
8. A binary predicate symbol $\subseteq$ whose both arguments are *group-subjects.*
9. Logical connectives and punctuations: as usual, including equality.

For instance, a fact that a subject S has access right R for object O is represented using a ground atom $s\text{-}holds(S, R, O)$. The group membership is represented as follows: for example, "a subject S is a member of G" is represented using the formula $S \in G$. We can also represent inclusion relationships between subject groups such as $G_1 \subseteq G_2$. Furthermore, we can represent constraints among subjects' authorizations. For example, the rule stating that for any subject and object, if the subject is the owner of this object, then the subject should have read and write rights for that object. This constraint can be represented as follows:

$$\forall so.s\text{-}holds(s, Own, o) \supset s\text{-}holds(s, Read, o) \wedge s\text{-}holds(s, Write, o).$$

On the other hand, suppose we have a constraint stating that for any subject s and group subject g, if s is a member of g, then s should have all the access rights that g has. This is the so-called *inheritance* property of authorizations. This constraint can be captured using the following formula:

$$\forall sgao.s \in g \wedge g\text{-}holds(g, a, o) \supset s\text{-}holds(s, a, o).$$

The following example shows how the traditional access matrix can be represented by a set of ground atoms.

Example 1 Consider the access matrix in Figure 1, where

	O1	O2	O3
S1	R, W	E, W	
S2		E, W	R

Figure 1. An access matrix.

R, W, E represent the rights of $Read$, $Write$ and $Execute$ respectively. This access matrix can be represented in our formalism as the following set of atoms.

$$\{s\text{-}holds(S_1, Read, O_1),\\ s\text{-}holds(S_1, Write, O_1),\\ s\text{-}holds(S_1, Write, O_2),\\ s\text{-}holds(S_1, Execute, O_2),\\ s\text{-}holds(S_2, Write, O_2),\\ s\text{-}holds(S_2, Execute, O_2),\\ s\text{-}holds(S_2, Read, O_3)\}.$$

2.2 The Policy Base

Let us now consider a formal definition of the policy base using the language $\mathcal{L}$. First we need to introduce some additional concepts that are relevant to this discussion.

Given the language $\mathcal{L}$ and a problem domain, we assume that the objects of the domain *are fixed*, and each constant of language $\mathcal{L}$ is mapped onto a specific object of the domain (different constants correspond to different objects). As we only consider finite domains, $\mathcal{L}$ will only contain a finite number of constants. Following the above assumption, in this paper we will not explicitly consider the creation and deletion of objects in the domain. We will show later how this issue is handled implicitly by inserting and deleting access right transformations.

The *Herbrand base* [4] of $\mathcal{L}$ is the set of all possible ground atoms of $\mathcal{L}$. A *Herbrand interpretation* of $\mathcal{L}$ is a subset of the Herbrand base of $\mathcal{L}$. Given a closed first order formula F of $\mathcal{L}$ and a Herbrand interpretation M of $\mathcal{L}$, we say that F *is satisfied* in M, denoted as $M \models F$, if F is *true* under the truth-value assignment defined by M. In this situation, we also say that M is a Herbrand model of F. For example, let $M = \{S \in G, \textit{g-holds}(G, Write, O), \textit{s-holds}(S, Write, O)\}$ and F be $\forall sgao.s \in g \wedge \textit{g-holds}(g, a, o) \supset \textit{s-holds}(s, a, o)$, then we have $M \models F$. If C is a set of closed first order formulas of $\mathcal{L}$, we say that C *is satisfied* in a Herbrand interpretation M of $\mathcal{L}$, denoted as $M \models C$, if each formula in C is satisfied in M.

The formal definition of our policy base is now given as follows:

Definition 1 *A* policy base PB *is a pair* (C, M) *where* C *is a finite set of closed first order formulas of* $\mathcal{L}$ *and* M *is a Herbrand model of* C. M *is also called a* state *of policy base* PB.

Intuitively, in a policy base $PB = (C, M)$, C represents the policy constraints about the domain of the system and M represents the agent's knowledge of access rights. For any ground atom F in M, we also use notation $PB \models F$.

Note that M only contains positive facts (ground atoms). Any fact not in M will be treated as negation of the fact. Therefore, if we have a policy base $PB = (\{\}, \{\textit{s-holds}(S, Read, O)\})$, and the domain further contains other objects S' and O', then PB actually also implies $\neg\textit{s-holds}(S', Read, O')$, $\neg\textit{s-holds}(S, Read, O')$, and $\neg\textit{s-holds}(S', Read, O)$. Therefore, we will use notation such as $PB \models \neg\textit{s-holds}(S', Read, O')$.

Example 2 Consider the following information and rules of access requirements from a mental health hospital [13].

1. Al is a patient.
2. Ed is a primary physician and he may read patients' records.
3. Jane is a primary physician.
4. Patients may not read their records.
5. The consulting physician may neither modify nor copy patients' records.
6. The primary physician may modify or copy patients' records.

Let P, CP, PP be group constants that represent group patient, group consulting-physician and group primary-physician respectively. *P-records* represents the patients' records object. The above information and rules can be represented as follows:

$$Al \in P \tag{1}$$

$$Ed \in PP, \textit{s-holds}(Ed, Read, \textit{P-records}) \tag{2}$$

$$Jane \in PP \tag{3}$$

$$\forall s.s \in P \supset \neg\textit{s-holds}(s, Read, \textit{P-records}). \tag{4}$$

$$\begin{aligned}\forall s.s \in CP \supset \neg\textit{s-holds}(s, Modify, \textit{P-records}) \wedge \\ \neg\textit{s-holds}(s, Copy, \textit{P-records}).\end{aligned} \tag{5}$$

$$\begin{aligned}\forall s.s \in PP \supset \textit{s-holds}(s, Modify, \textit{P-records}) \vee \\ \textit{s-holds}(s, Copy, \textit{P-records}).\end{aligned} \tag{6}$$

In our system, this can be specified as $PB = (\{(4), (5), (6)\}, \{(1), (2), (3)\})$, where $\{(4), (5), (6)\}$ is the set of constraints and $\{(1), (2), (3)\}$ is the set of facts.

Here are some important features of our policy base defined above.

1. Access constraints of the domain are explicitly represented in a policy base. Intuitively, these constraints characterize the basic access control properties that should be satisfied by any specific access rights instance of the system at any time. Hence, for a given system, at different points in time, the policy bases differ only due to a difference in the set of facts. For example, if $PB_1 = (C_1, M_1)$ and $PB_2 = (C_2, M_2)$ are the policy bases of a system at time t_1 and t_2 respectively, then the condition $C_1 = C_2$ should hold.

2. In this paper, denials of access rights are represented implicitly by their absence in the policy base. Since the agent usually only records the positive access rights facts in the system, anything s/he does not record is assumed to be denied by default. This is the so called *closed world assumption*. This helps us to give a complete semantics about the agent's knowledge of access rights in the system[1]. In fact, most real systems employ default properties in one way or another, e.g. restrictive or permissive access policies.

3 Policy Base Transformations

3.1 Definition of Transformation

Transformations change the state of the policy base. For a policy base $PB = (C, M)$, we view M as a set of *changeable* facts while C as a set of *non-changeable* formulas. Therefore during transformation, C always remains unchanged. We consider two basic types of transformations that can be performed on the policy base: addition of a new access right to the current policy base and deletion of a current access right from the policy base. Another type of transformation updating or modification can be represented by these two basic types of transformations. For instance, the effect of updating $s\text{-}holds(S_1, Read, O)$ to $s\text{-}holds(S_2, Read, O)$ can be seen to be equivalent to deleting $s\text{-}holds(S_1, Read, O)$ and then adding $s\text{-}holds(S_2, Read, O)$ in the policy base.

Definition 2 *A* transformation description $tran$

$[Pre(tran)|Post(tran)]$, *where*
$Pre(tran) = \{h_1, \cdots, h_m\}$,
$Post(tran) = \{l_1, \cdots, l_n\}$, *and*

h_i, l_j *(*$1 \leq i \leq m$, $1 \leq j \leq n$*) are ground literals of* $\mathcal{L}$.

Intuitively, $Pre(tran)$ represents the precondition of $tran$ in which every ground literal must be satisfied in the current policy base before $tran$ is performed, while $Post(tran)$ represents the postcondition of $tran$ in which every ground literal must be satisfied in the new policy base after $tran$ is performed. For instance, let $PB = (C, M)$ be a policy base and $tran = [\{s\text{-}holds(S, Read, O)\}|\{\neg s\text{-}holds(S, Read, O)\}]$ be a transformation description. The intuitive meaning of performing $tran$ on PB is that if $PB \models s\text{-}holds(S, Read, O)$ then after performing $tran$, in the resulting policy base PB', the condition $PB' \models \neg s\text{-}holds(S, Read, O)$ should hold. If $Pre(tran)$ is an empty set, then this means that there is no precondition for $tran$ to execute (i.e. $tran$ can always be executed). A transformation $tran$ is *executable* on a policy base PB if for every ground literal h in $Pre(tran)$, $PB \models h$[2]. We also denote $PB \models Pre(tran)$ (or $PB \models Post(tran)$) if $PB \models h$ for each h in $Pre(tran)$ (or $PB \models l$ for each l in $Post(tran)$).

Now we are ready to describe the transformation procedure formally. Let $PB = (C, M)$. The performance of a transformation $tran$ on the policy base PB is achieved based on the result of the transformation $tran$ on the state M of PB under the *principle of minimal change*. Informally, the minimal change principle states that during a state transformation, the difference between the initial state and the resulting state should be *as minimal as possible* under the restriction of policy constraints.

Let M_1 and M_2 be two Herbrand interpretations of language $\mathcal{L}$. $Diff(M_1, M_2)$ denotes the set of ground atoms such that any ground atom in $Diff(M_1, M_2)$ only occurs in one of M_1 and M_2.

For instance, let $M_1 = \{s\text{-}holds(S, Read, O), s\text{-}holds(S, Read, O')\}$ and $M_2 = \{s\text{-}holds(S, Read, O)\}$. Then $Diff(M_1, M_2) = \{s\text{-}holds(S, Read, O')\}$.

Definition 3 *Let* $PB = (C, M)$ *be a policy base,* $tran$ *a transformation description that is executable on* PB *(i.e.* $PB \models Pre(tran)$*). A Herbrand interpretation* M' *of* $\mathcal{L}$ *is called a* possible resulting state *after performing transformation* $tran$ *on* M *if and only if* M' *satisfies the following conditions:*

1. $M' \models C$ *and* $M' \models l$ *for every ground literal* l *in* $Post(tran)$.

2. *There does not exist other Herbrand interpretation* M'' *of* $\mathcal{L}$ *such that* M'' *satisfies Condition 1 and* $Diff(M, M'') \subset Diff(M, M')$.

A possible resulting policy base PB' *after performing transformation* $tran$ *on* PB *is then defined as* $PB' = (C, M')$.

Let us examine the above definition more closely. In order to perform the transformation on a policy base PB, we need to compute the transformation on the state of PB. Condition 1 states that the resulting state M' should satisfy the constraint(s) and the postcondition of $tran$, while condition 2 forces the change from M to M' to be as minimal as possible. Note that according to the above definition, the resulting policy base after performing a transformation may not be unique. Let us now consider a few examples.

[1] There are some limitations of the closed world assumption in reasoning about disjunctive information. This aspect is beyond the scope of this paper.

[2] Recall that $PB \models h$ if $h \in M$ and $PB \models \neg h$ if $h \notin M$.

3.2 Examples

We now give some examples to illustrate how transformations are performed using the model-based approach described above.

Example 3 Consider a policy base $PB = (C, M)$, where

$$\begin{aligned} M = \{&S_1 \in G, \textit{s-holds}(S_1, Write, O),\\ &\textit{s-holds}(S_1, Read, O),\\ &\textit{g-holds}(G, Read, O)\}, \text{ and}\\ C = \{&\forall s.s \in G \wedge \textit{g-holds}(G, Read, O) \supset\\ &\textit{s-holds}(s, Read, O)\}.\end{aligned}$$

where the constraint C says that if group G has the Read right for object O, then every member of G also has Read right for O. Now consider a transformation "if S_1 is a member of group G, then delete S_1's *Read* right for object O from PB". In our system, this transformation can be formally described as follows:

$$\begin{aligned} &tran = [Pre(tran) \mid Post(tran)], \text{ where}\\ &Pre(tran) = \{S_1 \in G, \textit{s-holds}(S_1, Read, O)\},\\ &\text{and}\\ &Post(tran) = \{\neg \textit{s-holds}(S_1, Read, O)\}.\end{aligned}$$

Clearly, $PB \models Pre(tran)$. So $tran$ is executable on PB. According to Definition 3, we have

$$M_1' = \{S_1 \in G, \textit{s-holds}(S_1, Write, O)\}.$$

$$\begin{aligned} M_2' = \{&\textit{s-holds}(S_1, Write, O),\\ &\textit{g-holds}(G, Read, O)\}.\end{aligned}$$

So, we get two possible resulting policy bases:

$$\begin{aligned} &PB' = (C, M_1') \text{ and } PB'' = (C, M_2') \text{ where}\\ &M_1' = \{S_1 \in G, \textit{s-holds}(S_1, Write, O)\}.\\ &M_2' = \{\textit{s-holds}(S_1, Write, O),\\ &\qquad \textit{g-holds}(G, Read, O)\}.\end{aligned}$$

Example 4 Let PB be the same as the previous example. Now consider the transformation "if S_1 is a member of group G, then change S_1's Write right for object O to Execute right". We can specify this transformation as follows:

$$\begin{aligned} &tran = [Pre(tran) \mid Post(tran)], \text{ where}\\ &Pre(tran) = \{S_1 \in G,\\ &\qquad \textit{s-holds}(S_1, Write, O)\},\\ &Post(tran) = \{\neg \textit{s-holds}(S_1, Write, O),\\ &\qquad \textit{s-holds}(S_1, Execute, O)\}.\end{aligned}$$

Clearly, $PB \models Pre(tran)$. So transformation $tran$ is executable on PB. It should be noted that since the term $Execute$ occurs in the postcondition of $tran$ (i.e. $\textit{s-holds}(S_1, Execute, O)$), it should also appear in the *Herbrand base* of our language $\mathcal{L}$ used in this example.

Using Definition 3, we have the following result:

$$\begin{aligned} M' = \{&S_1 \in G, \textit{s-holds}(S_1, Execute, O),\\ &\textit{s-holds}(S_1, Read, O),\\ &\textit{g-holds}(G, Read, O)\}.\end{aligned}$$

So, the final policy base is

$$\begin{aligned} &PB' = (C, M'), \text{ where}\\ &M' = \{S_1 \in G, \textit{s-holds}(S_1, Execute, O),\\ &\qquad \textit{s-holds}(S_1, Read, O),\\ &\qquad \textit{g-holds}(G, Read, O)\}.\end{aligned}$$

4 Conflict Resolution

4.1 The Problem

As the policy constraints are explicitly taken into account in our policy base, the transformations can be *nonmonotonic* in the sense that the addition of new access right(s) in the current policy base may also lead to a loss of some other access right(s). For instance, consider the following example.

Example 5 Let $PB = (C, M)$ be a policy base, where

$$\begin{aligned} M = \{&\textit{s-holds}(S_1, Read, O_1),\\ &\textit{s-holds}(S_2, Read, O_2)\}, \text{ and}\\ C = \{&\forall ao.\textit{s-holds}(S_1, a, o) \supset\\ &\neg \textit{s-holds}(S_2, a, o)\}.\end{aligned}$$

Suppose we want to add the fact $\textit{s-holds}(S_1, Read, O_2)$ into PB. Consider the constraint, $\textit{s-holds}(S_1, Read, O_2)$ implies $\neg\textit{s-holds}(S_2, Read, O_2)$. To maintain the consistency of PB, we need to either keep the policy base unchanged or remove $\textit{s-holds}(S_2, Read, O_2)$ and get a new PB where $M' = \{\textit{s-holds}(S_1, Read, O_1), \textit{s-holds}(S_1, Read, O_2)\}$.

¿From the above example we can see that by introducing policy constraints in our policy base, a transformation may cause some *indirect changes* to the policy base. However, in some situations, these indirect changes may lead to conflicts which need to be resolved. The approach described so far does not provide a resolution for conflicts. Let us consider the following example,

Example 6 Let $PB = (C, M)$ be a policy base, where

$$\begin{aligned} M = \{&S \in G\}, \text{ and}\\ C = \{&\forall sgo.s \in g \wedge \textit{g-holds}(g, Read, o) \supset\\ &\textit{s-holds}(s, Read, o)\}.\end{aligned}$$

Now consider the addition of $\textit{g-holds}(G, Read, FILE)$ to the policy base. According to Definition 3, there are two possible resulting states after performing such a transformation on M.

$$M'_1 = \{S \in G, g\text{-}holds(G, Read, FILE), \\ s\text{-}holds(S, Read, FILE)\}.$$
$$M'_2 = \{g\text{-}holds(G, Read, FILE)\}.$$

Let us examine the two possible resulting states M'_1 and M'_2 more closely. In this situation, since our policy constraint is equivalent to $g\text{-}holds(G, Read, FILE) \supset S \notin G \vee s\text{-}holds(S, Read, FILE)$, both M'_1 and M'_2 represent the minimal changes from M with respect to this particular transformation. M'_1 is obtained if the predicate $s\text{-}holds$ takes precedence over the predicate $\in$. This seems reasonable because after the addition of $g\text{-}holds(G, Read, FILE)$ into the policy base, S obtains a read right for $FILE$ because of the inheritance property. On the other hand, M'_2 is obtained if the predicate $\in$ takes precedence over the predicate $s\text{-}holds$. We have no reason to say that M'_2 is not reasonable according to our approach described in section 3.1. To resolve such issues, we need to specify preference ordering mechanisms on these predicates which reflect the security policies of the organizations.

4.2 An Approach

Several approaches to conflict resolution have been proposed. Strong and weak authorizations have been proposed in the Orion authorization model [2]. The basic idea behind this approach is that strong authorizations cannot be overridden, while weak authorizations can be overridden by strong or other weak authorizations, according to specified rules. Lunt [5] discussed the *most-specific rule* and *denials take precedence* approaches. Most-specific rule requires that if an individual subject is specifically granted or denied authorization for an object, this takes precedence over any other authorizations for the object that are granted or denied to groups to which the subject belongs. With denials take precedence approach, a subject or group's denial of authorization for an object takes precedence over any authorizations that the subject or group may have been granted for the object. We will use the approach of weak and strong authorization together with a preference ordering mechanism to resolve conflicts. We assign the newly added authorization(s) to be strong and the previously existing authorization(s) to be weak. Therefor, in our transformation, the newly added authorization(s) always override the existing authorization(s).

As described in section 2.1, there are four predicates $\in$, $\subseteq$, $s\text{-}holds$ and $g\text{-}holds$ in our language $\mathcal{L}$. We introduce the following preference ordering among the predicates in $\mathcal{L}$. We assign $g\text{-}holds$ to have a higher precedence than $\in$ and $\subseteq$, $\in$ and $\subseteq$ to have a higher precedence than $s\text{-}holds$. Formally, a strict partial ordering $\prec$ (i.e. antireflexive, antisymmetric and transitive) among predicates $\in$, $\subseteq$, $s\text{-}holds$ and $g\text{-}holds$ is defined as $s\text{-}holds \prec \in \prec g\text{-}holds$ and $s\text{-}holds \prec \subseteq \prec g\text{-}holds$. We can now extend our model-based transformation given in Definition 3 as follows. First we need to introduce some additional notations. Let M, M' be two Herbrand interpretations of $\mathcal{L}$. $M[g\text{-}holds]$, $M[s\text{-}holds]$, $M[\in]$ and $M[\subseteq]$ denote the set of all interpretations of predicates $g\text{-}holds$, $s\text{-}holds$, $\in$ and $\subseteq$ in M respectively. For example, if a Herbrand interpretation is

$$M = \{s\text{-}holds(S_1, A_1, O_1), \\ s\text{-}holds(S_2, A_2, O_2), \\ g\text{-}holds(G_1, A_3, O_3), \\ S_1 \in G_1, S_2 \in G_2, G_1 \subseteq G_2\},$$

then we have

$$M[g\text{-}holds] = \{g\text{-}holds(G_1, A_3, O_3)\},$$
$$M[s\text{-}holds] = \{s\text{-}holds(S_1, A_1, O_1), \\ s\text{-}holds(S_2, A_2, O_2)\},$$
$$M[\in] = \{S_1 \in G_1, S_2 \in G_2\}, \text{ and}$$
$$M[\subseteq] = \{G_1 \subseteq G_2\}.$$

On the other hand, $Diff_{g\text{-}holds}(M, M')$, $Diff_{s\text{-}holds}(M, M')$, $Diff_{\in}(M, M')$ and $Diff_{\subseteq}(M, M')$ denote the set of different interpretations on predicates $g\text{-}holds$, $s\text{-}holds$, $\in$ and $\subseteq$ in M and M' respectively. For instance, if

$$M' = \{s\text{-}holds(S_2, A_2, O_2), S_2 \in G_2, \\ G_1 \subseteq G_2\},$$

then

$$Diff_{g\text{-}holds}(M, M') = \{g\text{-}holds(G_1, A_3, O_3)\},$$
$$Diff_{s\text{-}holds}(M, M') = \{s\text{-}holds(S_1, A_1, O_1)\},$$
$$Diff_{\in}(M, M') = \{S_1 \in G_1\}, \text{ and}$$
$$Diff_{\subseteq}(M, M') = \{\}.$$

The following is the formal definition of the extended model-based transformation based on the preference ordering $\prec$.

Definition 4 *Let $PB = (C, M)$ be a policy base, $tran$ a transformation description that is executable on PB (i.e. $PB \models Pre(tran)$), and M a possible state of PB. A Herbrand interpretation M' of $\mathcal{L}$ is called a* possible resulting state *after transformation $tran$ on M based on the preference ordering $\prec$ (called $\prec$-transformation $tran$ on M), if and only if M' satisfies the following conditions:*

1. *$M' \models C$ and $M' \models l$ for every ground literal l in $Post(tran)$.*
2. *There does not exist other Herbrand interpretation M'' of $\mathcal{L}$ such that*
   (a) M'' satisfies Condition 1;
   (b) $Diff_{g\text{-}holds}(M, M'') \subset Diff_{g\text{-}holds}(M, M')$; or

(c) $M'[g\text{-}holds] = M''[g\text{-}holds]$ *and* $Diff_{\in}(M, M'') \subset Diff_{\in}(M, M')$ *or* $Diff_{\subseteq}(M, M'') \subset Diff_{\subseteq}(M, M')$*; or*

(d) $M'[g\text{-}holds] = M''[g\text{-}holds]$ *and* $M'[\in] = M''[\in]$ *and* $M'[\subseteq] = M''[\subseteq]$ *and* $Diff_{s\text{-}holds}(M, M'') \subset Diff_{s\text{-}holds}(M, M')$.

The following example shows how a transformation is performed using the extended model-based transformation based on the preference ordering $\prec$.

Example 7 A policy base $PB = (C, M)$, where

$$\begin{aligned} M = \{&S_1 \in G, S \in G, S_2 \in G_1, S \in G_1,\\ &g\text{-}holds(G, Read, O),\\ &s\text{-}holds(S, Read, O),\\ &s\text{-}holds(S_1, Read, O),\\ &g\text{-}holds(G_1, Execute, O),\\ &s\text{-}holds(S, Execute, O),\\ &s\text{-}holds(S_2, Execute, O)\}, \text{ and}\\ C = \{&\forall sgao.s \in g \wedge g\text{-}holds(g, a, o) \supset\\ &s\text{-}holds(s, a, o)\}. \end{aligned}$$

Consider the transformation "the members of group G cannot have execute right for O". That is, the addition of $\neg s\text{-}holds(S, Execute, O)$ and $\neg s\text{-}holds(S_1, Execute, O)$ to PB. Formally, the transformation can be represented as follows:

$$\begin{aligned} &tran = [Pre(tran) \mid Post(tran)], \text{ where}\\ &Pre(tran) = \{\}, \text{ and}\\ &Post(tran) = \{\neg s\text{-}holds(S, Execute, O),\\ &\quad \neg s\text{-}holds(S_1, Execute, O)\}. \end{aligned}$$

Since S belongs to both groups G and G_1 and using the above constraint, S inherits the access rights from both groups. That is, S holds *Read* and *Execute* rights for O. The transformation will change the access rights of S and S_1. The result of the transformation is that S and S_1 cannot have *Execute* right for O, which conflicts with the access right S has inherited from group G_1. To resolve this conflict, we use the preference ordering. ¿From the policy constraint, we get $\neg s\text{-}holds(S, Execute, O) \supset S \notin G_1 \vee \neg g\text{-}holds(G_1, Execute, O)$. Since $\neg s\text{-}holds(S, Execute, O)$ must be satisfied in the resulting policy base PB', this leads to S being removed from G_1's membership or $g\text{-}holds(G_1, Execute, O)$ being removed from the policy base. Since we defined the preference ordering to be $\in \prec g\text{-}holds$, $S \in G_1$ will be removed from the policy base.

Formally from Definition 4, we have

$$\begin{aligned} M' = \{&S_1 \in G, S \in G, S_2 \in G_1,\\ &g\text{-}holds(G, Read, O),\\ &g\text{-}holds(G_1, Execute, O),\\ &s\text{-}holds(S, Read, O),\\ &s\text{-}holds(S_1, Read, O),\\ &s\text{-}holds(S_2, Execute, O)\}. \end{aligned}$$

Our transformation is based on the model of PB. For a subject, its access rights, inherited from all of the groups that it belongs to, are within the same model(s). When performing transformations, the consistency of the model will guarantee the consistency of every related group. So we do not need to check the individual group for maintaining the consistency of PB.

So, the resulting policy base is:

$$\begin{aligned} &PB' = (C, M') \text{ where}\\ &M' = \{S_1 \in G, S \in G, S_2 \in G_1,\\ &\quad g\text{-}holds(G, Read, O),\\ &\quad g\text{-}holds(G_1, Execute, O),\\ &\quad s\text{-}holds(S, Read, O),\\ &\quad s\text{-}holds(S_1, Read, O),\\ &\quad s\text{-}holds(S_2, Execute, O)\}. \end{aligned}$$

5 Implementation Issues

In this section, we discuss the implementation of our extended model-based transformation approach. Our method is similar to that of Winslett's update theory [3, 11] in the sense of model-based semantics of change, but differs from it due to different ontologies.

5.1 The Basic Idea

Let $PB = (C, M)$ be a policy base. A transformation *tran* adds a set of literals $N = \{l_1, l_2, ..., l_n\}$ to PB. ¿From Definition 4, the resulting policy base is $PB' = (C, M')$ where $M' \models C$ and $M' \models N$. So M' is one of the Herbrand models of $N \cup C$ which has minimal difference from M with respect to the preference ordering $\prec$ defined in Condition 2 of Definition 4. Therefore, we have the following

$$M' = Min(M, Models(N \cup C), \prec)$$

Our implementation comprises the following stages:

Step 1: Generate the set of models of $N \cup C$: $Models(N \cup C)$.

Step 2: For each $M' \in Models(N \cup C)$, compute $Diff(M, M')$. Then $M' = Min(M, M', \prec)$.

Step 1 involves a procedure for model generation. In step 2, to obtain the resulting model, we need to compute $Diff(M, M')$ for each M' in $Models(N \cup C)$ and then find $Min(M, M', \prec)$.

5.2 The Algorithms

5.2.1 The Model Generator

For a policy base $PB = (C, M)$, generally the constraints in C may include universally quantified variables[3]. From the implementation point of view, we need to *ground* each constraint containing variables in C to all of its propositional instances[4]. For instance, in Example 3, C contains one constraint:

$$\forall s.s \in G \wedge g\text{-}holds(G, Read, O) \supset s\text{-}holds(s, Read, O)\}.$$

During implementation, this constraint needs to be replaced by its ground instance:

$$S_1 \in G \wedge g\text{-}holds(G, Read, O) \supset s\text{-}holds(S_1, Read, O).$$

Therefore, in the rest of this section, when we refer to a policy base $PB = (C, M)$, we assume that C only contains constraints without variable occurrence.

We need to define some additional concepts that will be used in our algorithms. For a policy base $PB = (C, M)$, an *inconsistency* is a set of literals whose conjunction is inconsistent with C. A *minimal inconsistency* is an inconsistency which has no subset that is also an inconsistency.

Let L be the set of all ground literals of the language defined in section 2.1. To get the set of models of $N \cup C$, first we need to find out the set $\mathcal{I}$ of minimal inconsistencies between L and C. This can be achieved using an inference engine. Given L and C, we can use the resolution proof to find out all the minimum-length proofs which lead to empty clauses; the required inconsistencies can then be directly read off from these proofs.

For instance, let us consider Example 3 and see how one can obtain the set $\mathcal{I}$ of minimal inconsistencies. To simplify the problem, let a stand for $S_1 \in G$, b for *s-holds*$(S_1, Write, O)$, c for *s-holds*$(S_1, Read, O)$ and d for *g-holds*$(G, Read, O)$. Then $N = \{\neg c\}$ and $C = \{a \wedge d \supset c\}$. Furthermore, $a \wedge d \supset c$ is equivalent to $\neg a \vee \neg d \vee c$. Here $L = \{a, \neg a, b, \neg b, c, \neg c, d, \neg d\}$. The resolution proof of Figure 2 shows the procedures needed to obtain the set $\mathcal{I} = \{a, d, \neg c\}$ of minimal inconsistencies.

Once the set $\mathcal{I}$ of minimal inconsistencies between L and C is obtained, a model M' of $N \cup C$ can be achieved using a maximal subset of L which contains N but does not contain any minimal inconsistency.

For the above example, considering the inconsistency $\{a, d, \neg c\}$, we get models $\{a, b\}$ and $\{b, d\}$. They are: $\{S_1 \in G, s\text{-}holds(S_1, Write, O)\}$ and $\{s\text{-}holds(S_1, Write, O), g\text{-}holds(G, Read, O)\}$.

[3] Technically, an existential quantifier in a formula can be eliminated by introducing Skolem function [4].

[4] This technique is often used in the implementation of first order dynamic systems, eg.[11].

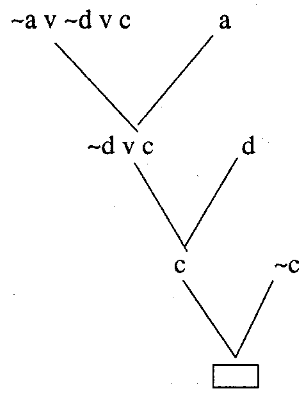

Figure 2. Resolution Proof.

The algorithm of model generator for $N \cup C$ is as follows:

Algorithm 1.
Input: A finite set N of ground literals, a finite set C of ground formulas and a finite set L of ground literals over $M \cup C$.
Output: A finite set of models of $N \cup C$.

1. Use resolution proof to find the set $\mathcal{I}$ of minimal inconsistencies between L and C.
2. Find the maximal subset of L which contains N but does not contain any inconsistency of $\mathcal{I}$.
3. From the set of all such maximal subsets of L form the set of models of $N \cup C$, i.e., *Models*$(N \cup C)$.

In the above algorithm, we achieve step 1 using a theorem prover[5]. In fact, step 1 can be pre-computed as a separate procedure for finding the set of minimal inconsistencies between L and C.

5.2.2 Resulting Model Finder

Given model M and the set of models of $N \cup C$, the resulting model finder obtains a corresponding $M' \in Models(N \cup C)$ for M under the measure of minimal change from M to M'. The model obtained is the resulting model. The algorithm is as follows:

Algorithm 2.
Input: A finite set of model M and a finite sets of models: Models$(N \cup C)$.
Output: A resulting model.

[5] Refer to [8] for detailed information on the theorem prover.

1. For every $M' \in Models(N \cup C)$, compute $Diff_{g\text{-}holds}(M, M')$, $Diff_{\in}(M, M')$, $Diff_{\subseteq}(M, M')$ and $Diff_{s\text{-}holds}(M, M')$.
2. Find the M' which has smallest set of $Diff(M, M')$, this M' is closest to M under the condition 2 of Definition 4.
3. The M' obtained is the resulting model.

6 Related Work and Discussion

In this paper, we have developed a logic based approach to formalize authorization policies and to describe nonmonotonic transformation procedures. Constraints have been used to represent the basic properties of authorizations in a system. A model based semantics has been employed to formalize the transformations. We have shown that our model allows the transformation of both explicit and implicit authorizations. Furthermore by introducing preference ordering on predicates of the language, we have extended our model-based semantics to resolve conflicts during transformation of authorizations. We have also discussed the implementation aspects of our model. At present, we are implementing the approach described in this paper.

The property of nonmonotonic transformation of authorization policies was also addressed by Sandhu and Suri [9]. However, as we mentioned earlier, their method is procedure based; we have shown in our full version technical report that their system is a special case within our framework. It corresponds to the case where the set of constraints is empty.

Let us now return to Woo and Lam's previous work [12] on a logic based specification of policy bases. Woo and Lam use Reiter's default logic [7], where a policy base is represented by a default theory. Hence the basic unit of access right in their formalism is a rule of the form $f : f'/g$. For instance, a rule such as $Read(S, O_1) : Read(S, O_2)/Read(S, O_3)$ has the following intuitive meaning: if subject S can read object O_1 and it is consistent to assume that S can also read O_2, then the agent will conclude that S can read O_3. Therefore, a fact $s\text{-}holds(S, Read, O)$ in our formalism becomes a special case of the rule of the form $T : /s\text{-}holds(S, Read, O)$. So Woo and Lam's policy base has more expressiveness in representing the system's access rights. However, there are some unignorable limitations in their formulation. First, they do not consider the constraints in the access control of a system. Hence it is not clear how to judge whether a policy base is legal or illegal with respect to the system restrictions. Second, using default theory to represent policy base also inherits the critical semantics problem of default logic. That is, given a policy base, it may have more than one extension, and even worse, it may not have any extension at all. Finally, since not much work has been done on the issue of update on default theory, we believe that it would be difficult to consider the authorization transformation based on Woo and Lam's specification. In comparison, our logic based approach for authorization transformation has the following major advantages: (i) our policy base has a clear and complete semantics; (ii) model-based transformation is easy to understand while at the same time being powerful, for instance, the nonmonotonic property of authorization transformation can be captured in our framework; (iii) our approach is feasible for implementation.

However there are several areas where further research work is required; we are addressing the following at present. In this paper, we only considered the fact-based authorization transformation in which the constraints on the access rights of the system remained fixed. We are currently extending this approach to include updating the constraints on the access rights of the system; we refer to this as the rule-based authorization transformation. That is, from a policy base $PB_1 = (C_1, M_1)$, after a rule-based authorization transformation, we obtain a new policy base $PB_2 = (C_2, M_2)$, where C_1 may differ from C_2. We are also investigating a way of proposing a minimal change principle to capture this kind of transformation.

References

[1] Yun Bai, Vijay Varadharajan, A Logic Based Approach for Transformation of Authorization Policies. Manuscript, September, 1996.

[2] F.Rabitti et al, A Model of Authorization for Next Generation Database Systems", ACM Trans on Database Systems, 1, March 1991.

[3] T.S-C. Chou, M. Winslett, Immortal: a Model-based Belief Revision System, *The 2nd International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufman Publishers Inc. pp 99–110, 1991.

[4] S.K. Das, *Deductive Databases and Logic Programming*, Addison-Wesley Publishing Company, UK, 1992

[5] T.F. Lunt, Discretionary Security for Object-Oriented Database Systems, Technical Report 7543, Computer Science Laboratory, SRI International, 1990.

[6] W.W. McCune, Automated Discovery of New Axiomatizations of the Left Group and Right Group Calculi, *Journal of Automated Reasoning*. 09(1): pp 1–24, 1992

[7] R. Reiter, A logic for default reasoning, *Artificial Intelligence*, 13(1-2): 81-132, 1980.

[8] S. Russell and P. Norrig, *Artificial Intelligence - A Modern Approach.* Prentice Hall, 1995..

[9] R.S. Sandhu and S. Ganta, On the Expressive Power of the Unary Transformation Model, *Third European Symposium on Research in Computer Security*, pp 301–318, 1994.

[10] R.S. Sandhu and G.S. Suri, Non-monotonic transformation of access rights, *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp 148–161, 1992.

[11] M. Winslett, *Updating Logical Databases*. Cambridge University Press, New York, 1990.

[12] T.Y.C. Woo and S.S. Lam, Authorization in distributed systems: A formal approach, *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp 33–50, 1992.

[13] V. Varadharajan and C. Calvelli, An access control model and its use in representing mental health application access policy, *IEEE Trans. Knowledge And Data Engineering, vol.8, no.1,* pp 81–95, 1996.