December 1997

A Formal Security Design Approach for Transformation of Authorizations in Information Systems

Yun Bai
University of Western Sydney

Vijay Varadharajan
University of Western Sydney

A Formal Security Design Approach for Transformation of Authorizations in Information Systems

Yun Bai & Vijay Varadharajan
Department of Computing
University of Western Sydney
P. O. Box 10
Kingswood, NSW 2747, Australia
Email {ybai, vijay}@st.nepean.uws.edu.au

Key words: information security, authorization policy, logic based specification, transformations

Executive Summary
In a multi-user, information-sharing system, authorization policy provides the ability to limit and control access to the system. In the real world, authorization policies need to capture the changing needs of applications. Representation, evaluation and analysis of such changes form an important part of the design of authorization policies. These changes are implemented via transformation of authorization policies. In this paper, we propose a logic based approach to specify authorization policies and to reason about transformation of authorization policies. In our system, the authorization policy is specified using a policy base which comprises a finite set of facts and a finite set of access constraints. The facts represent explicitly the access rights the subjects hold for the objects. The access constraints, on the other hand, are rules which the authorization policy should satisfy. We define the structure of the policy transformation and employ a model-based semantics to perform the transformation under the principle of minimal change. With the model-based approach, the transformation of a policy base is not based on the formulas presented in the policy base but on the individual model of the policy base. The principle of minimal change is used to guarantee that the change of the policy base after the transformation is as minimum as possible. Furthermore, we extend model-based semantics by introducing preference ordering to resolve possible conflicts during transformation. Our system is able to represent both implicit and incomplete authorization requirements and reason about nonmonotonic properties.

Abstract
In this paper, we propose a logic based approach to specify authorization policies and to reason about transformation of 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. Our system is able to represent both implicit and incomplete authorization requirements and reason about nonmonotonic properties.

1 Introduction

In a multi-user, information-sharing system, authorization service provides the ability to limit and control access to systems, applications and information, and to limit what entities can do with the information. In the real world, authorization policies need to capture the *changing* needs 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 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 and the mechanisms for reasoning about nonmonotonic properties.

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 objects. The access constraints, on the other hand, are rules which the authorization policy should satisfy.

The constraints also imply other facts which can be deduced from the policy base and the inheritance property of the policy base.

We first define the structure of the policy transformation. The structure describes the preconditions that need to be satisfied before the transformation and the postconditions after the transformation. Then we employ *a model-based semantics* (Winslett 1990) in the computation of the transformation; with the model based approach, the transformation of a policy base is not based on the formulas presented in the policy base but on the individual model of the policy base. The *principle of minimal change* (Winslett 1990) is used to guarantee that the change of the policy base after the transformation is as minimum as possible. We also introduce preference ordering in model based semantics to resolve possible conflicts.

Let us briefly mention some of the related work in this area. The work by Woo and Lam (1992) 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, Ganta and Suri (1994, 1992) addressed the issue of transformation of access rights. Their work was based on the access matrix, in which only explicit authorizations can be represented. They used a procedure-based approach to represent the transformation. Our work uses a logic approach and uses *a model based semantics* to formalize the transformation; furthermore both explicit and implicit authorizations can be represented in 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 authorizations. Section 3 defines the transformation description, discusses the computation of the transformation and its nonmonotonic property. Section 4 extends the model-based transformation introduced in section 3 by combining a preference ordering in it to resolve transformation conflicts. Finally, section 5 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 for our policy base model.

2.1 The language

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

1. Sort *subject*: with subject constants S, S_1, S_2 ..., and subject variables s, s_1, s_2, ...
2. Sort *group-subject*: with group subject constants G, G_1, G_2, ..., and group subject variables g, g_1 ,g_2,....
3. Sort *access-right*: with access right constants A, A_1, A_2,.... and access right variables a, a, a_2, ...
4. Sort *group-access-right*: with group access right constants GA, GA_1, GA_2, ..., and group access right variables ga, ga_1, ga_2, ...
5. Sort *object*: with object constants O, O_1, O_2, ..., and object variables o, o_1, o_2, ...
6. Sort *group-object*: with group object constants GO, GO_1, GO_2, ..., and group object variables go, go_1, go_2, ...
7. A ternary predicate symbol *s-holds* which takes arguments as *subject, access-right or group-access-right* and *object or group-object* respectively.
8. A ternary predicate symbol *g-holds* which takes arguments as *group-subject, access-right or group-access-right* and *object or group-object* respectively.
9. A binary predicate symbol $\in$ which takes arguments as *subject* and *group-subject* or *access-right* and *group-access-right* or *object* and *group-object* respectively.
10. A binary predicate symbol $\subseteq$ whose both arguments are *group-subjects, group-access-rights* or *group-objects*.
11. Logical connectives and punctuations: as usual, including equality.

In our language, a subject S has access right R for object O is represented using a ground formula *s-holds(S,R,O)*. The traditional representation of access matrix can be easily represented by a set of ground formulas as the following example shows.

Example 1 Consider the access matrix in Figure 1, where R, W, E represent the rights of

	O_1	O_2	O_3
S_1	R, W	E, W	
S_2		E, W	R

Figure 1: An access matrix.

Read, *Write* and *Execute* respectively. Clearly, this access matrix can be represented in our formalism as the following set of formulas.

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

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 $G_1 \subseteq G_2$ or between access right groups such as $GA_1 \subseteq GA_2$. Furthermore, we can represent constraints among subjects' authorizations. For example, 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:

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

2.2 The Policy Base

Using the language L, we can now give a formal definition of the policy base.

Definition 1 A *policy base PB* is a pair of *(F, C)* where F is a finite set of ground literal and C is a finite set of closed first order formulas.

In a policy base *PB=(F, C)*, F represents the agent's knowledge of access rights and C represents the policy constraints about the domain of the system.

A *model* of a policy base is the assignment of a truth value to every formula of the policy base in such a way that all formulas of the policy base are satisfied (Das 1992). Formally, we give the following definition.

Definition 2 A *model* of a policy base *PB=(F,C)* is defined to be a Herbrand model (Das 1992) of $F \cup C$. *PB* is said to be *consistent* if there exists some model of *PB*. The set of all models of *PB* is denoted as *Models(PB)*. A formula ψ is a *consequence* of *PB*, denoted as *PB* $\models \psi$, if $F \cup C \models \psi$. In this case, we also say ψ is *satisfied* in *PB*.

Example 2 Consider a policy base *PB=(F,C)*, where
F={s-holds(S_1,Read,O),s-holds(S_2,Read,O)}, and
C={s-holds(S_1,Read,O) $\supset \neg$*s-holds(S_2,Read,O)}*.
Clearly this policy base is not consistent as there does not exist a model for *PB*.

Example 3 Let the policy base be *PB=(F,C)*, where

F={s-holds(S,Read,O$_1$)},

C={s-holds(S,Read,O$_1$) É *s-holds(S,Read,O$_2$)* Ú *s-holds(S,Read,O$_3$)}*.

PB has three models m_1, m_2, m_3 as follows:

m_1 = *{s-holds(S,Read,O$_1$), s-holds(S,Read,O$_2$)}*,

m_2 = *{s-holds(S,Read,O$_1$), s-holds(S,Read,O$_3$)}*, *and*

m_3 = *{s-holds(S,Read,O$_1$), s-holds(S,Read,O$_2$), s-holds(S,Read,O$_3$)}*.

Intuitively, a policy base represents the agent's information about authorizations of the system and this information can be *incomplete* in the sense that some fact(s) may neither be explicitly represented in the policy base nor be consequence(s) of the policy base. On the other hand, the set of all models of a policy base represents all possible current states of the policy base and these states are complete. Therefore, we refer to a model in *Models(PB)* as a *possible state* of *PB*. Here are some of the important features of our policy base:

1. Representing implicit information about access rights. In the above examples, the access rights represented by the facts are explicit. The access rights deduced from the facts and constraints are implicit .

2. Representing inheritance of access rights. In example 2, the access rights *s-holds(S$_1$,Read,O$_1$) and s-holds(S$_2$,Read,O$_1$)* are Representing implicit information about access rights. In the above examples, the access rights represented by the facts are explicit. The access rights deduced from the facts and constraints inherited since both S_1 and S_2 are members of the group *g*, and from the constraint in *PB*, they inherit all the access rights that group *g* has.

3. Representing incomplete information about access rights. We adopt the *open world assumption* (Winslett 1990) in our policy base. The ground formulas which are neither explicitly nor implicitly specified in our policy base can be either true or false. That is, if *s-holds(S,R,O)* is not present and it cannot be deduced from the policy base, *S* may or may not hold the access right *R* for *O*. It is important to be able to allow the incompleteness of a policy base because the agent may not have complete information about the system's authorizations at the beginning; some access rights may arise later when other transformations are performed.

3 The Transformations

Transformations change the state of the policy base. For a policy base *PB=(F,C)*, we view *F* as a set of *changeable* literal while *C* as a set of *non-changeable* formulas. Therefore during transformation, *C* is always kept unchanged. We consider three basic types of transformations that can be performed on the policy base: addition of a new access right to the current policy base, deletion of a current access right from the policy base and update or modification of an access right in the policy base. The third type of transformation can be represented using the previous two types of transformations. For instance, the effect of updating *s-holds(S,Write,O)* to *s-holds(S$_1$,Write,O)* can be viewed to be equivalent to deleting *s-holds(S,Write,O)* and adding *s-holds(S$_1$,Write,O)* in the policy base. Furthermore, in our system, deleting an access right from the current policy base is represented by the addition of the negation of such an access right to the policy base. Figure 2 shows the basic outline of a transformation on a policy base.

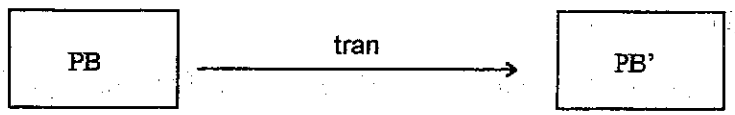

Figure 2: A transformation *tran* on *PB*.

Definition 3 A transformation description *tran* is a structure of the form

[Pre(*tran*)|Post(*tran*)], where

Pre(*tran*)={h_1,...,h_m},

Post(*tran*)={l_1,...,l_n}, and

h_i, l_j ($1 \le i \le m$, $1 \le j \le n$) are ground literal of 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. 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). We say a transformation *tran* is *executable* on a policy base *PB* if for every ground literal h in Pre(*tran*), $PB \models h$. 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. Under the model-based paradigm, the semantics of a transformation on a policy base is based not on the formulas presented in the policy base, but on the individual model of the policy base. A transformation is applied to each model individually. That is, the performance of a transformation *tran* on the policy base *PB* is achieved based on the result of the transformation *tran* on *every* possible state of *PB* under the *principle of minimal change*. Informally, the minimal change principle says 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 $PB=(F,C)$ be a policy base and $m_1, m_2 \in Models(PB)$. $Diff(m_1, m_2)$ denotes the set of ground atoms such that any ground atom only occurs in one of m_1 and m_2. For instance, consider example 3 presented in section 2,

$Diff(m_1,m_2)=\{s\text{-}holds(S,Read,O_2), s\text{-}holds(S,Read,O_3)\}$,
$Diff(m_1,m_3)=\{s\text{-}holds(S,Read,O_3)\}$, and
$Diff(m_2,m_3)=\{s\text{-}holds(S,Read,O_2)\}$.

Definition 4 Let $PB=(F,C)$ be a policy base, *tran* a transformation description that is executable on *PB*, and *m* a possible state of *PB*. A Herbrand interpretation m' of L is called a *possible resulting state* by performing *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 L such that m'' satisfies Condition 1 and $Diff(m,m'') \subset Diff(m,m')$.

We denote all such possible resulting states as $Res(m,tran)$. A policy base $PB'=(F', C)$ is called the *resulting policy base* by performing transformation *tran* on *PB* if and only if

$$Models(F' \cup C) = \bigcup_{m \in Models(PB)} Res(m,tran). \qquad (1)$$

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 every possible 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 between *m* and m' to be as minimal as possible. (1) shows that the resulting policy base *PB'* after performing transformation *tran* on *PB* is based on *Res(m, tran)* for every *m* in *Models(PB)*.

We now give an example to illustrate how transformation is performed using the model-based approach described above.

Example 4 A policy base $PB=(F,C)$, where
$F=\{S_1 \in G, S_2 \in G, s\text{-}holds(S_1,Write, O), g\text{-}holds(G,Read,O)\}$, and
$C=\{\forall s.s \in G \wedge g\text{-}holds(G,Read,O) \supset s\text{-}holds(s,Read,O)\}$,

Now consider the transformation that "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:

$tran=[Pre(tran)|\ Post(tran)]$, where
$Pre(tran)=\{S_1 \in G,\ s\text{-}holds(S_1,Write,O)\}$,
$Post(tran)=\{\neg s\text{-}holds(S_1,Write,O),\ s\text{-}holds(S_1,Execute,O)\}$

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*, it should also appear in the *Herbrand base* of our language L used in this example. If a ground literal is in the *Herbrand base* of a policy base but not in the policy base either explicitly or implicitly, this means that the truth value of this ground literal could be either positive or negative. Therefore our PB should have the following two Herbrand models:

$m_1 = \{S_1 \in G,\ S_2 \in G,\ s\text{-}holds(S_1,\ Write,\ O),\ g\text{-}holds(G,Read,O)$
$s\text{-}holds(S_1,Read,O),\ s\text{-}holds(S_2,Read,O)\}$, *and*
$m_2=\{S_1 \in G,\ S_2 \in G,\ s\text{-}holds(S_1,Write,O),\ g\text{-}holds(G,Read,O)$
$s\text{-}holds(S_1,Read,O),\ s\text{-}holds(S_2,Read,O),\ s\text{-}holds(S_1,Execute,O)\}$.

Following Definition 4, we have the following result:

$Res(m_1,tran)=\{m'\}$, *and*
$Res(m_2,tran)=\{m'\}$, *where*
$m'=\{S_1 \in G,\ S_2 \in G,\ s\text{-}holds(S_1,Execute,O),$
$s\text{-}holds(S_1,Read,O),\ s\text{-}holds(S_2,Read,O),\ g\text{-}holds(G,Read,O)\}$

As *Herbrand model* only describes positive literal, the newly added literal $\neg s\text{-}holds(S_1,Write,O)$ does not appear in m' but appears in the resulting PB'. From (1), we obtain the resulting policy base as follows:

$PB'=(F',\ C)$, *where*
$F'=\{S_1 \in G,\ S_2 \in G,\ \neg s\text{-}holds(S_1,Write,O),$
$s\text{-}holds(S_1,Execute,O),\ g\text{-}holds(G,Read,O)\}$.

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 $PB=(F,C)$ be a policy base, where
$F=\{S \in G,\ \neg s\text{-}holds(S,Read,FILE),\ \neg g\text{-}holds(G,Read,FILE)\}$, *and*
$C=\{\forall sgo:\ s \in g \wedge g\text{-}holds(g,Read,o) \supset s\text{-}holds(s,Read,o)\}$.
Now consider the addition of $g\text{-}holds(G,Read,FILE)$ into the policy base. Obviously PB has a unique model $m=\{S \in G\}$. Therefore, according to Definition 4, 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. Since in this situation, our policy constraint is equivalent to *g-holds(G,Read,FILE)* $\supset S \notin G \vee$ *s-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 by the idea of that the predicate *s-holds* is preferred to change comparing with the predicate $\in$. It seems that this is a reasonable resulting state saying that after the addition of *g-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 by the idea of that the predicate $\in$ is preferred to change comparing with the predicate *s-holds*. We have no reason to say that m_2 is not reasonable according to our approach described in section 3.1. The question is: which state do we prefer? And why? To solve this conflict, we need a preference ordering on these predicates.

4.2 The Approach

Several approaches to conflict resolution have been proposed. Castano et al (1994) use the concept of strong and weak authorizations. 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 (1990) discussed the *most-specific rule* and *denials take precedence* approaches.

We will use the approach of weak and strong authorization and together with preference ordering to resolve conflicts. We assign the newly added authorization(s) to be strong and the previous existing authorization(s) to be weak.

As described in section 2.1, there are four predicates $\in$, $\subseteq$, *s-holds* and *g-holds* in our language L. We assign *g-holds* has a higher precedence than $\in$ and $\subseteq$, $\in$ and $\subseteq$ have a higher precedence than *s-holds*. This can be achieved by introducing a preference ordering among these predicates in L. By combining such ordering in our model-based semantics, we can then provide a formal solution to the problem of conflict as we just mentioned above.

Formally, a strict partial ordering $<$ (i.e. antireflexive, antisymmetric and transitive) among predicates $\in$, $\subseteq$, *s-holds* and *g-holds* is defined as *s-holds* $< \in <$ *g-holds* and *s-holds* $< \subseteq <$ *g-holds*. Based on this idea, we can extend our model-based transformation defined by Definition 5. We first introduce some useful notations. Let m, m' be two Herbrand interpretations of L. m[g-holds], m[s-holds], m[$\in$] and m[$\subseteq$] denote the set of all interpretations of predicates *g-holds*, *s-holds*, $\in$ and $\subseteq$ in m respectively. For example, if a Herbrand interpretation is

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

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

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

then

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

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

Definition 5 Let $PB=(F,C)$ be a policy base, *tran* a transformation description that is executable on *PB*, and m a possible state of *PB*. A Herbrand interpretation m' of L is called a *possible resulting state* by performing transformation *tran* on m based on the preference ordering < , 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 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
   $Diff_{s\text{-}holds}(m,m'') \subset Diff_{s\text{-}holds}(m,m')$.

We denote all such possible resulting states based on the preference ordering < as $Res^{<}(m,tran)$. A policy base $PB'=(F', C)$ is called the *resulting policy base* by performing <-transformation *tran* on *PB* if and only if

$$Models(F' \cup C) = \bigcup_{m \in Models(PB)} Res^{<}(m,tran). \qquad (2)$$

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

Example 6 A policy base $PB=(F,C)$, where

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

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, it can be represented as follows:

tran=[Pre(tran)|Post(tran)], where
Pre(tran)={}, and
Post(tran)={$\neg s\text{-}holds(S, Execute,O)$, $\neg s\text{-}holds(S_1, Execute,O)$*}.*

PB has only one model:

$m=\{S_1 \in G, S \in G, S_2 \in G_1, S \in G_1$, *g-holds(G,Read,O), s-holds(S,Read,O), s-holds*(S_1,*Read,O), g-holds*(G_1,*Execute,O), s-holds(S,Execute,O), s-holds*(S_2,*Execute,O)*$\}$.

Since S belongs to both groups G and G_1 and using the constraint, it 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 inherited from group G_1. Since our conflict resolution policy is that the newly added facts override the previously existing facts, then this will force S to lose the *execute* right for O. But our constraint says that if S is a member of G_1, S holds the access right(s) that G_1 holds, that is, S holds *Execute* right for O. This again results in a conflict. As constraints need to be always true, this leads to S being removed from G's membership or *g-holds*(G_1,*Execute,O)* being removed from the policy base. Since we defined the preference ordering $\in <$ *g-holds*, $S \in G_1$ will be removed from the policy base.

Formally from Definition 5, we have

Res(m,tran)=$\{m'\}$, where

$m'=\{S_1 \in G, S \in G, S_2 \in G_1$, *g-holds(G,Read,O), g-holds*(G_1,*Execute,O), s-holds(S,Read,O), s-holds*(S_1,*Read,O), s-holds*(S_2,*Execute,O)*$\}$.

Our transformation is based on the model(s) 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(s) will guarantee the consistency of every related group. So we do not need to check the individual group for maintaining the consistency of *PB*.

From (2), the resulting policy base is as follows:

PB'=(F', C), where

$F'=\{S_1 \in G, S \in G, S_2 \in G_1, S \notin G_1$, *g-holds*($G_1$,*Execute,O), g-holds(G,Read,O),* $\neg$*s-holds(S,Execute,O),* $\neg$*s-holds*(S_1,*Execute,O)*$\}$.

5 Conclusion

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 implicit information and the inheritance property of authorizations. A model based semantics is employed to formalize the transformation. As it has been showed, our model allows the representation of both explicit and implicit authorizations. Furthermore by introducing preference ordering on predicates of the language, we extended our model-based semantics to resolve the conflicts during a transformation of authorizations. We also considered the implementation issues of our model in details in (Bai and Varadharajan 1996). Currently, a project towards an implementation of this approach is being undertaken.

Due to the space limitation, in this paper we have not discussed other's work on the issue of logic-based transformation of authorizations. However, to the best of our knowledge, our work presented in this paper is one of the most original work by using a model-based semantics to formalize transformation of authorization from a logical point of view. We expect that our system provides a unified framework which can be used to model other methods. In fact, in a full version of this paper (Bai and Varadharajan 1996), we have shown that Sandhu and Ganta's non-monotonic transformation system can be subsumed by our system.

References

Bai, Y and Varadharajan, V. "A Logic Based Approach for Transformation of Authorization Policies." Manuscript, September, 1996.

Castano, S.; Fugini, M.; Martella, G. and Samarati, P, "Database Security." Addison-Wesley Publishing Company, UK, 1994

Das, S.K, "Deductive Databases and Logic Programming." Addison-Wesley Publishing Company, UK, 1992

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

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

Sandhu, R.S, and Suri, G.S, "Non-monotonic transformation of access rights." Proceedings of IEEE Symposium on research in Security and Privacy, pp 148–161, 1992.

Winslett, M, "Updating Logical Databases." Cambridge University Press, New York, 1990.

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