Regular Paper

Access Control Policy Analysis Using Free Variable Tableaux

Hiroaki Kamoda,† Masaki Yamaoka,† Shigeyuki Matsuda,†
Krysia Broda†† and Morris Sloman††

The specification of access control policies for large, multi-organization applications is difficult and error-prone. Sophisticated policies are needed for fine-grained control of access to large numbers of entities, resulting in many policies specified by different security administrators. Techniques such as role based access control (RBAC) have been proposed to group policies and provide a framework for inheriting policies based on role hierarchies. RBAC does not prevent inconsistencies and conflicts arising in the policy specifications, though, which can lead to information leaks or prevent required access. This paper proposes an approach using free variable tableaux to detect conflicts and redundant policies resulting from the combination of various types of authorization and constraint policies. This approach uses static analysis to enable complete detection of modality and static constraint policy conflicts.

1. Introduction

In many organizations, the only form of authorization policy specification is very low level access control lists in workstations and servers, network-layer packet filtering rules in firewalls, or possibly an access control matrix defined simply by a combination of subjects and targets [25]. These may be physically distributed throughout the organization and are difficult to analyze. Large-scale inter-organizational applications require more sophisticated approaches to the specification of security policies to cater for very large numbers of subject and target entities. Role based access control (RBAC) [9] is being used as a means of grouping policies and using inheritance to simplify the specification. An object-oriented language such as Ponder [8] can be used to specify both management and authorization aspects of security policy. XACML [21] with the RBAC profile [21] and X-RBAC [4] have also been proposed to allow easy transfer of policies between systems. Standard RBAC has some limitations in that it supports only a role hierarchical structure for subjects and defines only positive authorization policies. However, XACML with the RBAC profile or Ponder has a more flexible role concept which supports negative authorizations as well as obligation policies for performing security related actions, such as raising events or logging access. It also permits hierarchical structuring of targets. In addition, there are formal extensions to the RBAC model; for example, Kalam, et al. [14] defined the ORBAC model and Bertino, et al. [3] defined the TRBAC model.

As the overall structure of the access control model and policy description languages become more complex, though, it is increasingly difficult for a system administrator to verify that the policies meet the application requirements. For instance, the RBAC concept of policy inheritance means that policies specified for one role may be inherited by other roles according to the role-hierarchy. In general, an access control policy is defined in terms of three elements — subject, target, and action. Naturally, the access control model should support the propagation of policies according to the structural hierarchy of targets and actions as well as subjects. This simplifies policy specification but makes the policy analysis more difficult, increasing the risk that an administrator will mistakenly define conflicting and redundant policies. A conflicting policy may result in information leakage or prevent required access, while redundant policies complicate policy management and reduce performance. To avoid these problems, both conflicting and redundant policies should be detected and corrected.

There are several approaches to conflict detection in the literature. For example, Graham, et al. [11] proposed a method to detect a modality conflict by using a decision table. Strembeck [27] presented a method to detect a static separation of duty conflict caused by propagation. However, these methods do not address conflicts caused by the structure of actions and they are very specific to a par-

† NTT DATA CORPORATION
†† Imperial College London

ticular policy model. They cannot be easily extended for a variety of policy specification methods [6),13),22),24)], nor do they give an administrator information about the causes of conflicts. To verify and correct the policy, both detection and identification of the specific policies causing the conflict are needed.

In this paper, we describe a method to detect conflicting and redundant policies by translating a policy specification notation such as Ponder into a formal first-order logic for analysis using the well known free variable tableaux to statically detect conflicting and redundant policies. We assume an access control model in which subjects, targets, and actions all have some structure, but the approach is general enough to be applied to other policy notations or access control models. Our method can statically detect modality conflicts typically arising from explicit positive and negative authorizations referring to the same subject, target, or action. Modality conflicts may also arise implicitly from policies propagating within the subject, target, or action hierarchy. In addition, policies may conflict with overall application constraints such as requirements for the separation of duties between roles. Using abductive inference, our method can also give us useful information for correcting policy conflicts. Moreover, redundant policies can be detected through the same approach.

The paper is organized as follows: Section 2 introduces the various types of policy and conflict examples that are discussed in this paper. Section 3 presents an outline of the conflict and redundancy detection method using free variable tableaux; in Section 4 we illustrate the formalization of policies; in Section 5 we describe the method to detect conflicting policies and prove the completeness of the method, and in Section 6 we describe the method to detect redundant policies; in Section 7 we discuss the performance of our approach and in Section 8 we present some extensions of the approach. Conclusions and future work are presented in Section 9.

2. Policy Analysis Framework

Policy specification and analysis is needed for many complex inter-organizational applications. An example is the on-demand virtual private network (VPN) framework for healthcare [12),29)]. In this framework, the decision as to whether a user can connect to a VPN between medical devices or institutions is based on a policy which must be flexible enough to cope with changes in the network topology and the complex interactions between multiple organizations involved in healthcare. Moreover, fine-grained access control is needed to distinguish between access to medical records, administrative records, and applications such as remote diagnosis. Traditional VPN-based access control only operates at the network layer, whereas our approach supports more sophisticated application-oriented access control defined in terms of subject and target role structures. Obligation policies specify management actions to be taken with respect to security. Policies propagate up or down the role structures as explained in Section 4.5. Policies may be defined in terms of composite actions requiring a number of sub-actions, which can also result in conflicts if separate policies are defined for the sub-actions as explained below. Moreover, Chinese wall policies and separation of duty policies may be needed. In this section, examples of these policies and the types of conflict which may arise are briefly explained.

The most basic policies used in the framework are authorization and obligation policies as indicated below. Obligation policies are event-condition-action rules.

$$\begin{aligned} \text{r1} &: \texttt{Auth+}(S_8, T_5, A_7) \\ \text{r2} &: \texttt{Auth-}(S_2, T_5, A_7) \\ \text{r3} &: \texttt{Obli+}(E_1, S_3, T_2, A_8) \\ \text{r4} &: \texttt{Obli-}(E_2, S_2, T_2, A_7) \end{aligned}$$

where S_2-S_8, T_2-T_5, and A_7-A_8 are the subject roles, target roles, and actions shown in **Fig. 1** and **Fig. 2**. Policies r1 and r2 respectively state that a subject role *clinical staff* is allowed to

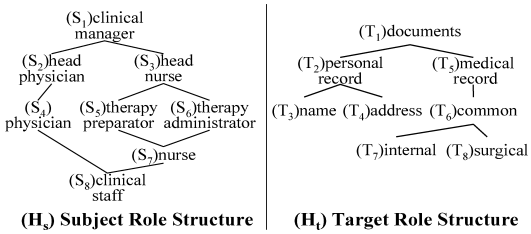

Fig. 1 Examples of role structures.

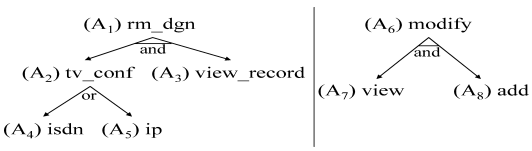

Fig. 2 Examples of action compositions.

view a target role *medical record* and that a subject role *head physician* is not permitted to *view* a target role *medical record*. Policies r3 and r4 respectively state that a subject role *head nurse* must *modify* a target role *personal record* if event E_1 (e.g., new address notification) is received and a subject role *head physician* must not *view* a target role *personal record* if event E_2 happens (e.g., if it is Mondays because the system will be heavily loaded). *Explicit modality conflicts* may occur between positive and negative polices defined for the same subject roles, target roles, actions, and events.

Typically, positive authorizations propagate up the subject role hierarchy and negative authorizations propagate down, as specified by a propagation policy:

$$\text{pr1} : \texttt{prop}(\texttt{Auth}-, \mathcal{H}_{\texttt{a}}, \texttt{DOWN})$$

This policy specifies that a negative authorization policy which applies to a high level role of the subject role hierarchy $\mathcal{H}_a$, defined in Fig. 1, should propagate downwards to lower level roles. The following authorization policies can thus be derived from policy r2.

$$\begin{aligned} \text{r1\_1} &: \texttt{Auth}-(S_4, T_5, A_7) \\ \text{r1\_2} &: \texttt{Auth}-(S_8, T_5, A_7) \end{aligned}$$

Clearly, policy r1_2 derived from propagation policy pr1 has an *implicit modality conflict* with policy r1, as there are both positive and negative authorizations for the same subject, target, and action.

Action composition indicates the dependence relationships between actions as shown in Fig. 2, where the left side defines the action composition policy:

$$\begin{aligned} \text{ac1} &: \texttt{rm\_dgn} = \texttt{tv\_conf} \wedge \texttt{view\_record} \\ \text{ac2} &: \texttt{tv\_conf} = \texttt{isdn} \vee \texttt{ip} \end{aligned}$$

Action composition policy ac1 states that remote diagnosis *rm_dgn* consists of two subactions *tv_conf* and *view_record*, both of which must be performed. Action composition policy ac2 specifies that at least one action *isdn* or *ip* is performed in the system when performing a TV conference `tv_conf`. These action composition policies may result in conflicts. For example, the following three authorization policies conflict.

$$\begin{aligned} \text{r5} &: \texttt{Auth}+(S_4, T_2, A_1) \\ \text{r6} &: \texttt{Auth}-(S_4, T_2, A_2) \\ \text{r7} &: \texttt{Auth}-(S_4, T_2, A_3) \end{aligned}$$

Policy r5 specifies *physician* is allowed to perform the action *rm_dgn*, while r6 and r7 specify that actions *tv_conf* and *view_record* are both prohibited. This sort of conflict is called a *constraint conflict* caused by action composition.

Another type of constraint conflict may result from a Chinese wall policy which constrains the number of target roles that can be accessed simultaneously. For example, the policy

$$\text{cw1} : \texttt{CW}(S_8, \{T_2, T_5\}_1, A_7)$$

specifies that a subject role *clinical staff* is allowed to *view* at most one of targets T_2 or T_5. This policy also results in a policy conflict:

$$\begin{aligned} \text{r8} &: \texttt{Auth}+(S_8, T_2, A_7) \\ \text{r9} &: \texttt{Auth}+(S_8, T_5, A_7) \end{aligned}$$

According to policies r8 and r9, subject role S_8 can view both targets, which conflicts with cw1.

A separation of duty policy is a policy that constrains the number of actions which can be performed simultaneously. For example, the policy

$$\text{sod1} : \texttt{SoD}(S_8, T_2, \{A_7, A_8, A_9\}_2)$$

specifies that at most two of actions A_7, A_8, or A_9 can be performed, which may also result in constraint conflicts.

As described in this section, several types of policy are used and several kinds of conflict may occur within the on-demand VPN framework. Rigorous definitions of the policies and conflicts are given in Section 4.

3. Tableaux Method

3.1 Overview

In this section we outline the use of *free variable tableaux* (FVT) [10] to detect conflicts and redundant policies.

Policy propagation up and down the role hierarchies derives additional implicit policies from a single original one. This simplifies policy specification, but it may cause an *implicit modality conflict*. Action composition policies may cause a form of constraint conflict. Using different algorithms to detect each type of conflict is inefficient and the system extensibility is reduced if a new conflict detection algorithm must be constructed for every new type of policy constraint.

The FVT method facilitates the detection of all types of conflict and redundant policies with a single algorithm, and also infers the causes of inconsistencies. The method is applied to logical sentences translated from the original policy specification notation so only a new translator is needed for a new policy notation.

Detection of a conflict effectively requires

that a contradiction $\perp$ be derived from a conjunction of policies Γ. To prove that C results from Γ (i.e., $\Gamma \models C$) is equivalent to showing that the set $\{\Gamma, \neg C\}$ is inconsistent (i.e., $\{\Gamma, \neg C\} \models \perp$). Using FVT, we can show not only whether a set of policies is conflicting, but also whether a policy can be deduced from the given set of policies. Furthermore, in many cases the method can determine that a set of policies is not in conflict, or that a policy cannot be deduced. This is further discussed in Section 7. The FVT method is a sound and complete theorem prover upon which can be built simple abductive reasoning. Moreover, it has optimized implementations. We outline the methods to detect conflicts and redundancies below.

Conflict Detection Two steps are needed to detect a conflict using FVT:

i) each policy $r_i \in \mathcal{P}$, where $\mathcal{P}$ stands for a set of policies, is translated into a logical sentence $\zeta(r_i)$.
ii) the FVT method is applied to sentences $\{\zeta(r_i)\}$ to detect any possible conflicts, by detecting any inconsistency, and to obtain information that shows the cause of the conflict.

Policy Deduction The following two steps are needed to verify that a policy r can be derived from a given set of policies.

i) each policy $r_i \in \mathcal{P}$ and the policy r that should be verified to be derivable from the set of policies $\mathcal{P}$ are translated into logical sentences $\zeta(r_i)$ and $\zeta(r)$.
ii) the FVT method is applied to the set of sentences $\{\{\zeta(r_i)\}, \neg\zeta(r)\}$ to verify whether policy r can be derived from other policies r_i by detecting inconsistency.

This kind of policy deduction can be used to check whether a certain policy may be derived from the given set of policies. These details are explained later.

As explained above, to perform both conflict detection and policy deduction, all we have to do is define the following translation mapping ζ from policies to logical sentences, such that conflicting policies become inconsistent sentences in logic.

$$\begin{array}{rccc} \zeta: & \mathcal{P} & \rightarrow & \mathcal{L} \\ & \cup & & \cup \\ & r & \mapsto & \zeta(r) \end{array}$$

where $\mathcal{P}$ is a set of policies and $\mathcal{L}$ is a set of sentences. The particular mapping chosen uses first-order logic and predicates P, O, and R to express, respectively, permission, obligation, and refrain. This allows both very natural translations and the use of standard and efficient first-order theorem provers; in contrast, the use of modal logic requires more complex theorem provers [19].

Once policies have been translated into logic, a conflicting policy is detected and a policy deduction is made in the same way despite the differences in the original policy descriptions. This means that our approach can easily be applied to various types of policy definition languages.

3.2 Why FVT for Policy Analysis?

Several automated theorem provers, such as OTTER [15),20)], PTTP [26)], Setheo [17)], and LeanTap [2)], use FVT. The advantages of FVT are that it is easy to implement and modify. Several implementations have been published [1),23)], showing that the method is comparable with others through analysis using TPTP [28)], a standard library of test problems for automated theorem proving systems. We choose FVT because a security policy conflict analysis system should use technology that is flexible enough to accommodate change and fast enough to analyze large numbers of policies.

3.3 Policy Conflict Analysis by FVT

The details of the FVT method have been explained by Fitting, et al. [10)]. Here, we briefly explain how the FVT method analyzes policies.

We use the [Type III] example from **Fig. 5** and show that three sentences (r17, r18, and Ax1) become conflicting. A tableau is developed as a tree, such that every piece of data is analyzed in every branch of the tree, unless a branch has already become conflicting. In a free variable tableau, if conflicting sentences can be made to appear on the same path the path is closed (indicated by a horizontal line in Fig. 5). If all paths are closed, then the given sentences are conflicting. The analysis starts from the premise that the data are not conflicting and shows that all possibilities resulting from the assumption lead to contradiction. A datum is analyzed by considering the possible truth values of its constituents. For example, a sentence of the form $A \rightarrow B$ is true if either $\neg A$ is true or B is true. This leads to two possibilities, represented in the tableau by two branches. Basic rules to build a tableau are shown in **Fig. 3**. A sentence $\forall x(E_x \rightarrow B)$ is true for each instance of variable x. In the FVT method, a free vari-

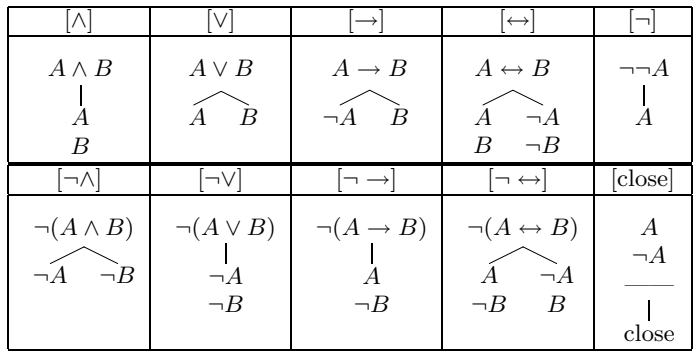

Fig. 3 Tableaux rules.

able is substituted for x, say x_1, to give the free variable instance $E_{x_1} \rightarrow B$, which is analyzed as above. That is, it is true if either $\neg E_{x_1}$ is true or B is true. In the first branch of Fig. 5, we can see that if E_C is true the branch will close. Abduction allows us to assume the occurrence of event E_C, which is then available as an assumption in the other branches. In particular, it allows for the second branch to be closed if x_1 is bound to C. In general, when bindings are made to variables they are propagated throughout the tableau. In the figures, results of this propagation are shown in brackets below the appropriate formulas. The third branch closes through the use of Ax1. The final outcome of the analysis is that if atom E_C becomes true, then there can be a conflict.

4. Policy Formalization

In this section we describe how policies can be formalized using the concepts introduced in Section 2.

4.1 Roles

A standard RBAC model has been proposed [9], and access control policies are often specified using roles. A role is a named collection of privileges. In the RBAC model, a role hierarchy related to subjects is defined and an obvious extension is to include target role structures. A partial order relation is defined among these roles and the graph representation of the relation is called a *role hierarchy*. Individual subjects and targets take on assigned roles. In particular, the role hierarchy corresponding to subjects is called a *subject role hierarchy* (SRH) and that corresponding to targets is called a *target role hierarchy* (TRH). Examples of these role structures are shown in Fig. 1. In addition to the policies, a role hierarchy must also be translated into logic for the purposes of the analysis. A role hierarchy is translated as indicated in **Fig. 4** (1), where $\mathcal{H}$ is a role hierarchy and $i \geq_{\mathcal{H}} j$ means i is greater than j in the partial order of $\mathcal{H}$. For example, the role hierarchy represented in Fig. 1 ($\mathcal{H}_s$) is translated as

$$\begin{aligned}\zeta(\mathcal{H}_s) \quad := \{ \quad & H_{\mathcal{H}_s}(S_1, S_2), H_{\mathcal{H}_s}(S_1, S_3), \\ & H_{\mathcal{H}_s}(S_2, S_4), H_{\mathcal{H}_s}(S_3, S_5) \\ & H_{\mathcal{H}_s}(S_3, S_6), H_{\mathcal{H}_s}(S_5, S_7) \\ & H_{\mathcal{H}_s}(S_6, S_7), H_{\mathcal{H}_s}(S_4, S_8) \\ & H_{\mathcal{H}_s}(S_7, S_8)\}\end{aligned}$$

where $H_{\mathcal{H}}(i, j)$ means that i is a *direct senior role* of j in role structure $\mathcal{H}$.

4.2 Authorization Policy

A positive authorization policy (`Auth+`) defines the action A_1 that a subject role S_1 is *permitted* to perform on a target role T_1. A negative authorization policy (`Auth-`) defines the action A_1 that a subject role S_1 is *forbidden* to perform on a target role T_1. For example, in XACML and Ponder, both authorization policies are defined as follows.

XACML:

```
<Rule RuleId="1"
Effect="Permit|Deny">
  <Subject> S1 </Subject>
  <Resource> T1 </Resource>
  <Action> A1 </Action>
</Rule>
```

Ponder:

```
type auth+|- RuleID_1
(subject S1, target T1) {
action A1;
}
```

Some details are omitted from the XACML for simplification and the following notation is used to describe an authorization policy in this paper.

$$\texttt{Auth}\pm(S_1,\ T_1,\ A_1).$$

(1)	$\zeta(\mathcal{H})$	:=	$\{H_{\mathcal{H}}(i,j) : i \geq_{\mathcal{H}} j\}$
(2)	$\zeta(\texttt{Auth+}(S_1,\ T_1,\ A_1))$	:=	$\forall x(E_x \rightarrow P(S_1,T_1,A_1))$
(3)	$\zeta(\texttt{Auth–}(S_1,\ T_1,\ A_1))$	:=	$\forall x(E_x \rightarrow \neg P(S_1,T_1,A_1))$
(4)	$\zeta(\texttt{Obli+}(E_1,S_1,\ T_1,\ A_1))$	:=	$E_1 \rightarrow O(S_1,T_1,A_1)$
(5)	$\zeta(\texttt{Obli–}(E_1,S_1,\ T_1,\ A_1))$	:=	$E_1 \rightarrow R(S_1,T_1,A_1)$
(6)	Ax1	:	$\forall s,t,a(O(s,t,a) \rightarrow P(s,t,a))$
(7)	Ax2	:	$\forall s,t,a(\neg(O(s,t,a) \wedge R(s,t,a)))$
(8)	$\zeta(\text{prop1}) = \zeta(\text{prop2})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(z,x) \rightarrow P(z,y,a))$
(9)	$\zeta(\text{prop3}) = \zeta(\text{prop4})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(x,z) \rightarrow P(z,y,a))$
(10)	$\zeta(\text{prop5}) = \zeta(\text{prop6})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(y,z) \rightarrow P(x,z,a))$
(11)	$\zeta(\text{prop7}) = \zeta(\text{prop8})$	:=	$\forall x,y,z,a(P(x,y,a) \wedge H_{\mathcal{H}}(z,y) \rightarrow P(x,z,a))$
(12)	$\zeta(A_1 = \Gamma(A_2,\cdots,A_n))$	:=	$\forall x,y(P(x,y,A_1) \leftrightarrow \Gamma(P(x,y,A_2),\cdots,P(x,y,A_n)))$
(13)	$\zeta(\text{cw})$	:=	$\forall x,y\left(\bigvee_{i=1}^{n!/(m!(n-m)!)}\left(\bigwedge_{j=1}^{n} P_{ij}(x,T_j,y)\right)\right)$
(14)	$\zeta(\text{sod})$	:=	$\forall x,y\left(\bigvee_{i=1}^{n!/(m!(n-m)!)}\left(\bigwedge_{j=1}^{n} P_{ij}(x,y,A_j)\right)\right)$
(13′)	$\zeta(\text{cw1})$	:=	$(\neg P(S,T_1,A) \wedge P(S,T_2,A))$
		$\vee$	$(P(S,T_1,A) \wedge \neg P(S,T_2,A))$
(14′)	$\zeta(\text{sod1})$	:=	$(\neg P(S,T,A_1) \wedge \neg P(S,T,A_2) \wedge P(S,T,A_3))$
		$\vee$	$(\neg P(S,T,A_1) \wedge P(S,T,A_2) \wedge \neg P(S,T,A_3))$
		$\vee$	$(P(S,T,A_1) \wedge \neg P(S,T,A_2) \wedge \neg P(S,T,A_3))$

Fig. 4 Definitions of mapping ζ.

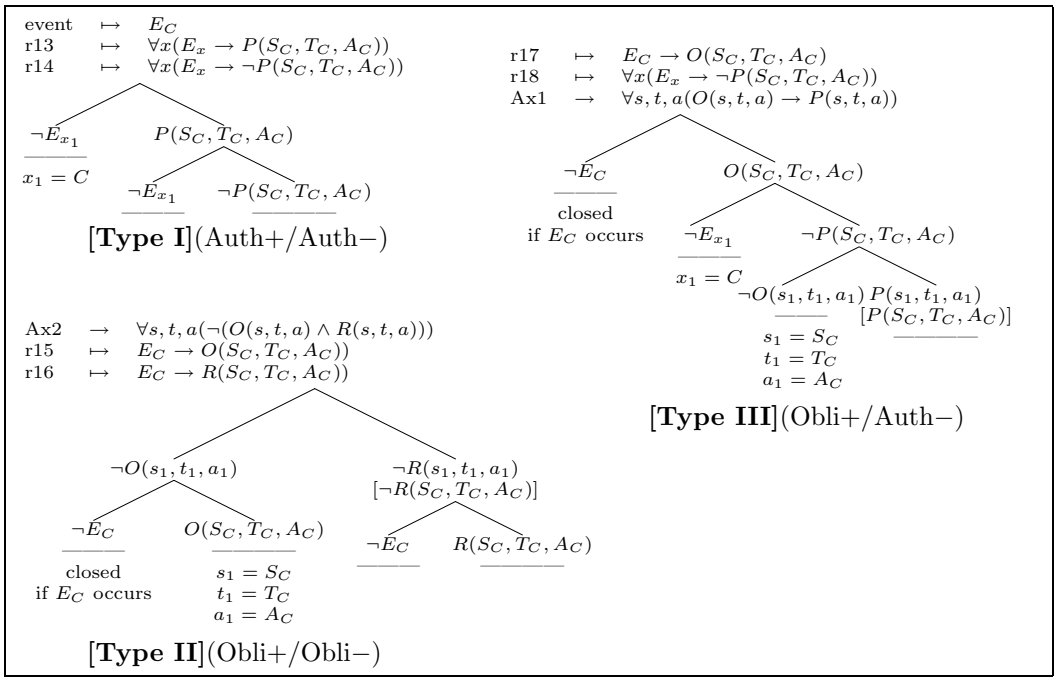

Fig. 5 Explicit modality conflict detection.

The mapping ζ of authorization policies is defined in translations (2) and (3) of Fig. 4. Predicate P can be read as "subject role S_1 is permitted to carry out action A_1 on target role T_1". Atom E_x says that event x occurs. Then, for example, the second translation can be read as "for any event E_x, S_1 is not permitted to carry out A_1 on T_1".

4.3 Obligation Policy

In XACML, only positive obligation is mentioned, while the syntax of obligation policy is not strictly defined. In Ponder, obligation policies are event-condition-action rules and both positive and negative obligation policies are de-

fined. A positive obligation policy (`Obli+`) defines action A_1 that subject role S_1 *must* perform on target role T_1 when event E_1 occurs. A negative obligation policy (`Obli-`) defines action A_1 that subject role S_1 *must not* perform on target role T_1 when event E_1 occurs. In Ponder a negative obligation policy is called a *refrain policy*. Examples of positive and negative obligation policies in Ponder are

Ponder:

```
type oblig RuleID_2
(subject S1, target T1) {
on E1;
do A1;
}
type refrain RuleID_3
(subject S1, target T1) {
on E1;
do A1;
}
```

These obligation policies are represented in this paper as

$$\texttt{Obli}\pm(E_1,\ S_1,\ T_1,\ A_1).$$

The translation mapping ζ of obligation policies is defined in (4) and (5) of Fig. 4. In these translations, the predicate O can be read as "subject role S_1 must carry out action A_1 on target role T_1" and R can be read as "subject role S_1 must not carry out action A_1 on target role T_1". Then, for example, the first translation can be read as "if event E_1 occurs then S_1 must carry out action A_1 on target role T_1".

4.4 Axiom

In addition to the transformation of authorization and obligation policies, there is a need for two axioms that relate P, O, and R; i.e., an obligation policy requires an authorization policy to permit the action and it contradicts a negative obligation policy. These axioms are described in (6) and (7) of Fig. 4. Ax1 is used to detect conflicts involving both authorization and obligation policies and Ax2 is used to detect conflicts between positive and negative obligation policies.

4.5 Propagation Policy

As shown in Section 4.2, an authorization policy is defined by using a role that has a partial order relation and a propagation policy defines how an authorization policy propagates in accordance with the partial order. In standard RBAC and Ponder, the direction of the propagation is always fixed, but we define an explicit propagation policy for each role structure which is more flexible than the implicit propagation in standard role hierarchies. This propagation policy is defined as

$$\texttt{prop}(\texttt{Auth}+|-, \texttt{SRH}|\texttt{TRH}, \texttt{Up}|\texttt{Down})$$

`SRH` and `TRH` respectively indicate the subject and target role structures to which the propagation policy is applied. `Up` and `Down` indicate the direction of the propagation, where `Up` means that the policy propagates upward through the partial order from the lowest element (most junior role) and `Down` means that the policy propagates downward from the highest element (most senior role).

The syntax of the propagation policy allows eight types of propagation policies to be defined:

$$\begin{array}{lcl}
\text{prop1} & : & \texttt{prop}(\texttt{Auth}+, \mathcal{H} \in \texttt{SRH}, \texttt{UP}) \\
\text{prop2} & : & \texttt{prop}(\texttt{Auth}-, \mathcal{H} \in \texttt{SRH}, \texttt{Down}) \\
\text{prop3} & : & \texttt{prop}(\texttt{Auth}+, \mathcal{H} \in \texttt{SRH}, \texttt{Down}) \\
\text{prop4} & : & \texttt{prop}(\texttt{Auth}-, \mathcal{H} \in \texttt{SRH}, \texttt{UP}) \\
\text{prop5} & : & \texttt{prop}(\texttt{Auth}+, \mathcal{H} \in \texttt{TRH}, \texttt{UP}) \\
\text{prop6} & : & \texttt{prop}(\texttt{Auth}-, \mathcal{H} \in \texttt{TRH}, \texttt{Down}) \\
\text{prop7} & : & \texttt{prop}(\texttt{Auth}+, \mathcal{H} \in \texttt{TRH}, \texttt{Down}) \\
\text{prop8} & : & \texttt{prop}(\texttt{Auth}-, \mathcal{H} \in \texttt{TRH}, \texttt{UP})
\end{array}$$

More than one propagation policy or no propagation policy can be defined as required. These eight propagation policies are translated into the four sentences defined as (8) to (11) in Fig. 4. From the fact that $(A \wedge B) \rightarrow C$ is equivalent to $(\neg C \wedge B) \rightarrow \neg A$, we can easily prove that, for example, the prop1 and prop2 policies are translated into the same sentence, as we likewise can for the other cases above.

4.6 Action Composition Policy

It is natural to regard actions, like roles, as having a structure similar to a role hierarchy; we call this an action composition policy. This simplifies policy definition as only the composite action needs to be specified. For example, three actions — A_1, A_2, and A_3 — may have an action composition policy defined as

$$A_1 = A_2 \wedge A_3$$

This means that performing an action A_1 is equivalent to performing actions A_2 and A_3. Thus, defining an authorization policy,

$$\text{r10} \ : \ \texttt{Auth}+(S_1, T_1, A_1)$$

is equivalent to defining the two authorization policies

$$\begin{array}{lcl}
\text{r11} & : & \texttt{Auth}+(S_1, T_1, A_2) \\
\text{r12} & : & \texttt{Auth}+(S_1, T_1, A_3)
\end{array}$$

The syntax of the action composition policy is defined by n actions $A_1, \cdots, A_n$:

$$A_1 = \Gamma(A_2, \cdots, A_n)$$

where Γ is a Boolean combination using $\wedge$, $\vee$, and $\neg$ of A_2, $\cdots$, A_n. The mapping ζ for the action composition policy is defined as (12) in Fig. 4.

4.7 Chinese Wall and the Separation of Duty Policies

A *Chinese wall policy* [5] and a *separation of duty policy* [7] respectively define the constraints for target roles and actions. A Chinese wall policy defines a constraint on the maximum size of a subset of target roles on which a subject role can perform an action. A separation of duty policy defines the maximum size of a subset of actions (usually one) that a subject role can perform on a target role. In this paper we discuss only static separation of duty and its conflicts, and do not consider dynamic separation of duty. The syntax of these policies is

$$\begin{array}{lcl} \text{cw} & : & \texttt{CW}(all, \{T_1, \cdots, T_n\}_m, all) \\ \text{sod} & : & \texttt{SoD}(all, all, \{A_1, \cdots, A_n\}_m), \end{array}$$

where m $(0 < m < n)$ is a cardinality for targets and actions. That is, the policy cw defines that any subject role can perform any action on up to m target roles of $\{T_1, \cdots, T_n\}_m$, and the policy sod defines that any subject role can perform up to m actions of $\{A_1, \cdots, A_n\}_m$ on any target role. The Chinese wall and separation of duty policies are formalized in (13) and (14) of Fig. 4, where $P_{ij}(\cdot) = P(\cdot)$ or $\neg P(\cdot)$, and $P(\cdot)$ and $\neg P(\cdot)$ must be carefully selected such that m "$P(\cdot)$"s and $n - m$ "$\neg P(\cdot)$"s appear in the formalization without omission and without redundancy. If a Chinese wall policy must be defined for a specific subject role or action in place of an arbitrary role, one can replace the arbitrary values x or y in the formalization of (13) in Fig. 4 with a specific subject role or action such as S_1 or A_1. To give an intuitive understanding of this principle, the formalizations of policies cw1 and sod1, as shown in Section 2, are respectively given in (13′) and (14′) of Fig. 4 .

5. Conflict Detection

Here, we show that the FVT can detect several types of policy conflict. First we define three types of conflict which may occur in our policy model — *explicit modality conflict*, *implicit modality conflict*, and *constraint conflict*. Then we argue for the completeness of our method by describing how FVT works. This means we prove that all kinds of conflict can be detected by using FVT.

5.1 Explicit Modality Conflict

Lupu, et al. [18] pointed out that a modality conflict may result from certain combinations of authorization and obligation policies when they refer to the same subject, target, and action; these combinations are {Auth+/Auth−}, {Obli+/Obli−}, and {Obli+/Auth−}. In this paper we classify modality conflicts into two types — *explicit* and *implicit*. An explicit modality conflict occurs between two explicitly defined authorization or obligation policies. An implicit modality conflict occurs as a result of the implicit derivation of one or more policies from a propagation policy.

Definition 1 (Explicit modality conflict). For an arbitrary constant C, three pairs of policies are defined as explicit modality conflicts:

[Type I] (Auth+/Auth− Conflict)

$$\begin{array}{lcl} \text{r13} & : & \texttt{Auth}+(S_C, T_C, A_C) \\ \text{r14} & : & \texttt{Auth}-(S_C, T_C, A_C) \end{array}$$

[Type II] (Obli+/Obli− Conflict)

$$\begin{array}{lcl} \text{r15} & : & \texttt{Obli}+(E_C, S_C, T_C, A_C) \\ \text{r16} & : & \texttt{Obli}-(E_C, S_C, T_C, A_C) \end{array}$$

[Type III] (Obli+/Auth− Conflict)

$$\begin{array}{lcl} \text{r17} & : & \texttt{Obli}+(E_C, S_C, T_C, A_C) \\ \text{r18} & : & \texttt{Auth}-(S_C, T_C, A_C) \end{array}$$

For these explicit modality conflicts, the following theorem holds.

Theorem 1 (Completeness of the explicit modality conflict detection). All explicit modality conflicts can be detected by the FVT method.

Proof. To prove Theorem 1, we apply the FVT method for each kind of conflicting pair defined in Definition 1. The policies need to be first translated into logic, using the mapping ζ defined in Fig. 4, and then the FVT method is applied. In Fig. 5 we show the translation and the result of analyzing these pairs by the FVT. Note that, strictly speaking, since the completeness of the FVT method itself has already been proved, it is enough to prove only the validity of the definition of mapping ζ as shown in Fig. 4. As shown in Section 3.3, it is proved that [Type III] results in closed tableaux. Since both [Type I] and [Type II] pairs also result in closed tableaux, as shown in Fig. 5, the completeness of the FVT method is proved for [Type I] and [Type II]. That is, the completeness of the FVT method for explicit modality conflicts has been proved. Note that in [Type I] a conflict only happens if an event occurs, so we assume an

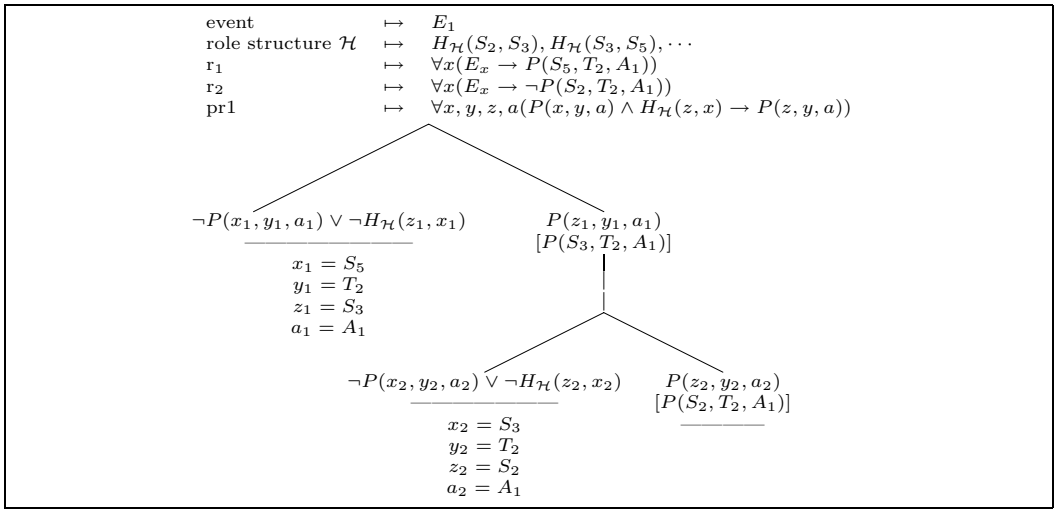

Fig. 6 Implicit modality conflict detection.

arbitrary event E_C occurs. □

5.2 Implicit Modality Conflict

Here, we describe the implicit modality conflicts caused by propagation policies. First, we show some examples of implicit modality conflicts and then prove that the FVT method can detect all implicit modality conflicts.

Definition 2 (Implicit modality conflict). For two arbitrary authorization policies,

$$\texttt{Auth}+(S_C^+, T_C^+, A_C^+),$$
$$\texttt{Auth}-(S_C^-, T_C^-, A_C^-),$$

let

$$\Omega_+ = \{\texttt{Auth}+(S_{C_i}^+, T_{C_i}^+, A_C^+)\}$$
$$\Omega_- = \{\texttt{Auth}-(S_{C_i}^-, T_{C_i}^-, A_C^-)\}$$

be the respective sets of authorization policies derived by propagation policies. The modality conflict that occurs between the policies in the set $\Omega_+ \cup \Omega_-$, or between a policy in $\Omega_+ \cup \Omega_-$ and another explicitly defined authorization or obligation policy is an implicit modality conflict. For this definition, the following theorem holds.

Theorem 2 (Completeness of the implicit modality conflict detection). All implicit modality conflicts can be detected by FVT.

Before proving this theorem, we show that FVT can detect an implicit modality conflict between r1, r2, and pr1 as described in Section 2. The result of analyzing these policies using FVT is shown in **Fig. 6**. Since a conflict only happens if an event occurs, we assume an arbitrary event E_1 occurs. To simplify the diagram, we omitted some details. For example, in the first branch of Fig. 6, if variables $\{x_1,\ y_1,\ z_1,\ a_1\}$ are given the values $\{S_2,\ T_2,\ S_3,\ A_1\}$, then the branch contradicts assumption $P_{\mathcal{R}}(S_2, S_3)$ and Policy r2. From the tableau, we deduce that these policies conflict with each other and that the conflict is due to the propagation $\{S_2, S_3, S_5\}$.

Proof. We give an outline of the proof here. We have already proved that every explicit modality conflict can be detected. This means that to prove Theorem 2 we need to show that every authorization policy involved in the set $\Omega_+ \cup \Omega_-$ in Definition 2 can be derived by the definition of mapping ζ defined in (1) and from (8) to (11) of Fig. 4. However, this is clear from the formalization. A strict proof of this would be provided by mathematical induction on the variables over the depth of the role hierarchies. □

5.3 Constraint Conflict

Here, we define the constraint conflicts caused by an action composition policy, the Chinese wall policy, and the separation of duty policy. We then prove that the FVT method can detect all kinds of constraint conflict.

5.3.1 Conflict Caused by Action Composition Policies

A conflict caused by an action composition policy is defined as follows.

Definition 3 (Constraint conflict caused by an action composition policy). Three types of conflict are caused by action composition policies. They are defined as follows, where, for example, Type IV defines that four

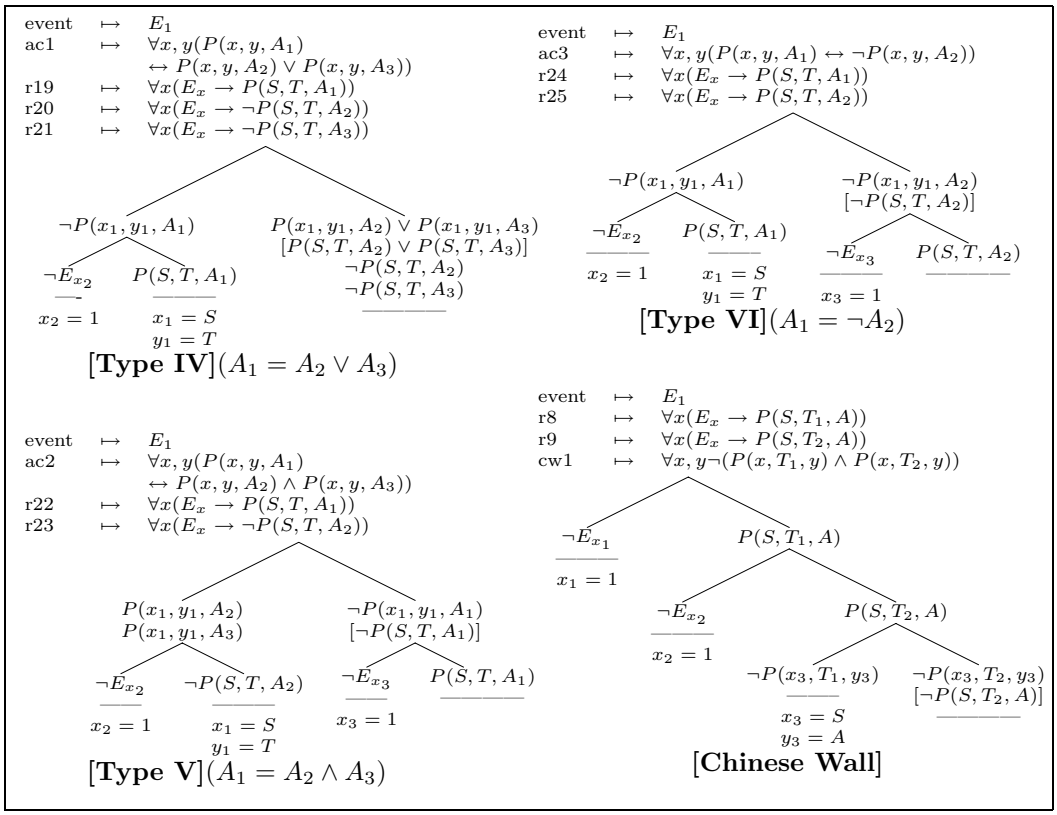

Fig. 7 Constraint conflict detection.

policies — ac1, r19, r20, and r21 — result in a conflict.

[Type IV]

ac1 : $A_1 = A_2 \vee A_3$
r19 : $\texttt{Auth+}(S_C, T_C, A_1)$
r20 : $\texttt{Auth-}(S_C, T_C, A_2)$
r21 : $\texttt{Auth-}(S_C, T_C, A_3)$

[Type V]

ac2 : $A_1 = A_2 \wedge A_3$
r22 : $\texttt{Auth+}(S_C, T_C, A_1)$
r23 : $\texttt{Auth-}(S_C, T_C, A_2)$ or $\texttt{Auth-}(S_C, T_C, A_3)$

[Type VI]

ac3 : $A_1 = \neg A_2$
r24 : $\texttt{Auth+}(S_C, T_C, A_1)$
r25 : $\texttt{Auth+}(S_C, T_C, A_2)$

where C is an arbitrary constant value and the `Auth+/-` policies may be explicit or implicit, meaning they can be given explicitly or derived by propagation. The conflicts caused by other types of action composition policies are defined inductively from these three definitions.

Theorem 3 (Completeness of the constraint conflict detection). All constraint conflicts caused by action composition policies can be detected by the FVT method.

Proof. We give an outline of the proof here. As all kinds of constraint conflict caused by action composition policies can be inductively defined from the three types — [Type IV], [Type V], and [Type VI] — all we have to do is prove that these three types become a conflict.

The result of analyzing these three types of policy using the FVT method is shown in [Type IV] to [Type VI] of **Fig. 7**. To simplify the diagram, we omitted some details. We can recognize that every type results in a conflict since all branches are closed. This means that Theorem 3 is true. □

5.3.2 Conflict Caused by Chinese Wall Policies

Here, we show that the FVT method can also detect a constraint conflict caused by a Chinese wall policy. In Fig. 7, [Chinese Wall] shows the result of applying the FVT to policies cw1, r8, and r9 described as examples in Section 2.

In general, the constraint conflict caused by a Chinese wall policy is defined as follows.

Definition 4 (Constraint conflict caused by a Chinese wall policy). The following policies Ω indicate the constraint conflict caused by a Chinese wall policy, where $1 \leq m \leq n$, $1 \leq k_i \leq n$, and $k_i \neq k_j$ $(i \neq j)$, and C is an arbitrary constant.

$$\begin{array}{lcl} \text{cw} & : & \texttt{CW}(all, \{T_1, T_2, \cdots, T_n\}_m, all) \\ \Omega & = & \{\texttt{Auth+}(S_C, T_{k_i}, A_C)\}_{i=1}^{m+1} \end{array}$$

where the `Auth+/-` policies may be explicit or implicit, meaning they can be given explicitly or derived by propagation.

That is, if more than m positive authorization policies are defined, this becomes a conflict. The following theorem holds for this definition.

Theorem 4 (Completeness of the constraint conflict detection). All constraint conflicts caused by Chinese wall policies can be detected by the FVT method.

Proof. From the premise of Definition 4, more than m positive authorization policies are defined for different T_{k_i}. By using the mapping ζ of (2) in Fig. 4, we get sentences including more than m positive literals of the form $P(S_C, T_{k_i}, A_C)$. According to (13) in Fig. 4, from policy cw we get the sentence

$$\bigvee_{i=1}^{n!/(m!(n-m)!)} \left(P_{i1}(x, T_1, y) \wedge \cdots \wedge P_{in}(x, T_n, y)\right).$$

This sentence involves m negative literals $\neg P(x, T_{k_i}, y)$ with at least one in each branch, thus causing all branches of the tableaux to be closed. Hence, when the mapping ζ is used, the set of rules defined in Definition 4 always conflict. □

Note that constraint conflicts caused by the separation of duty policy can also be defined in a similar way and all of them can be detected by the FVT method.

6. Policy Deduction

As described in Section 3, the FVT method is also applied to verify that a policy can be deduced from a given set of polices, and hence to detect a redundant policy. For example, given the policies

$$\begin{array}{lcl} \text{r26} & : & \texttt{prop}(\texttt{Auth+}, \mathcal{H} \in \texttt{SRH}, \texttt{UP}) \\ \text{r27} & : & \texttt{Auth+}(S_1, T, A) \\ \text{r28} & : & \texttt{Auth+}(S_2, T, A) \end{array}$$

we see that policy r27 is redundant since it can be derived from policies r26 and r28. While a redundant policy may not cause a serious problem, it could complicate policy administration and reduce performance so it should be eliminated.

To detect that policy r27 is redundant, the FVT method should be applied to the set $\{\zeta(\mathcal{H}), \zeta(r26), \zeta(r28), \neg\zeta(r27)\}$. If this set results in a closed tableau, it shows that policy r27 is redundant since the result indicates that $\zeta(r27)$ can be derived from the set $\{\zeta(\mathcal{H}), \zeta(r26), \zeta(r28)\}$.

In general, to verify that a certain policy r_i $(1 \leq i \leq n)$ is redundant for a given policy set $\{r_1, \cdots, r_n\}$, the FVT method can be applied for the set $\{\zeta(r_1), \cdots, \neg\zeta(r_i), \cdots, \zeta(r_n)\}$.

The same technique can be used to verify that a particular policy r not defined in the given policy set $\{r_1, \cdots, r_n\}$ can be derived from the policy set. The FVT method in this case is applied to the set $\{\zeta(r_1), \cdots, \zeta(r_n), \neg\zeta(r)\}$.

In this way, the FVT method can be used to detect a conflict as well as to detect a redundant policy using the same algorithm.

7. Computational Time

In this section, we discuss the computational time needed to analyze a set of policies. We have experimentally estimated the computational time by using leanCoP [23], which is an FVT implementation in Prolog based on model elimination tableaux [16]. Although the abduction function was not implemented, approximately estimating the computational time needed by our method was still useful.

In general, the computational time depends on the number of rules, the order in which the sentences are analyzed by tableaux, and the complexity of the sentences, especially for action composition. We estimated the computational time for the following cases.

Case I: (i) A set of policies containing an explicit modality conflict — {Auth+ / Auth−} (r13 and r14 described in Section 5.1) or {Obli+/Auth−} (r17 and r18 described in Section 5.1) — and other non-conflicting policies. (ii) A set of non-conflicting policies containing only authorization and obligation policies.

Case II: (i) A set of policies containing an implicit modality conflict — {Auth+ / Auth−} (r1, r2, pr1, and subject role structure $\mathcal{H}_s$ described in Section 2) — and other non-conflicting policies. (ii) A set of non-conflicting policies containing only authorization and propagation policies.

Table 1 Computer specifications.

CPU	Pentium 4, 3.20 GHz
Memory	1,024 MB
OS	Windows XP SP2
Prolog	SWI-Prolog Version 5.4.7

Case III: (i) A set of policies containing a constraint conflict caused by an action composition policy (ac1, r19, r20, and r21 described in Section 5.3.1) or by a Chinese wall policy (cw1, r8, and r9 described in Section 2), and other non-conflicting policies. (ii) A set of non-conflicting policies containing only authorization, action composition, and Chinese wall policies.

Case IV: (i) A set of mixed policies containing an implicit modality and constraint conflict caused by both subject and target propagation policies (prop1 and prop6 described in Section 4.5), authorization policies containing subjects and targets as described in Fig. 1, a Chinese wall policy (cw1 described in Section 2), and other non-conflicting policies. (ii) A set of non-conflicting policies containing authorization policies, propagation policies, and Chinese wall policies.

We measured the computational time for the above four cases with respect to the number of policies (2, 4, 8, 16, 32, 64, 128, 256, 512, 760, 1,024, 1,500, or 2,048). For each case, we prepared three policy sets in which the policies were differently ordered, and then we measured the average time until the analysis was completed. We also investigated the listing of conflicting policies to keep them separated by as far as possible so that we could measure the time for the situation that would be expected to take the longest time to analyze by tableau. **Table 1** shows the specifications of the computer used for the estimation. The results from both experiments were similar (**Fig. 8** to **Fig. 10**). In the graphs, the vertical axis shows the time needed to detect a conflict, described by a solid line, and the horizontal axis shows the number of policies simultaneously analyzed by the FVT method.

As Fig. 8 shows, an explicit modality conflict could be detected in a comparatively short time — about 25 to 40 seconds for 2,048 policies. This was because these types of conflict can be detected by simply comparing a set of policies. On the other hand, it took much longer to detect implicit modality conflicts (**Fig. 9**)

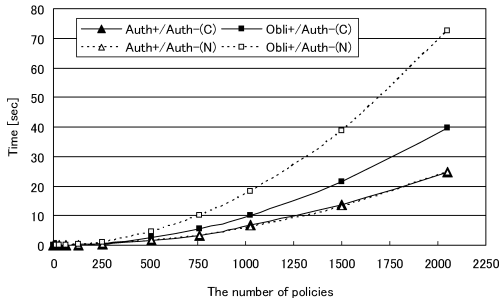

Fig. 8 Time needed to detect an explicit modality conflict.

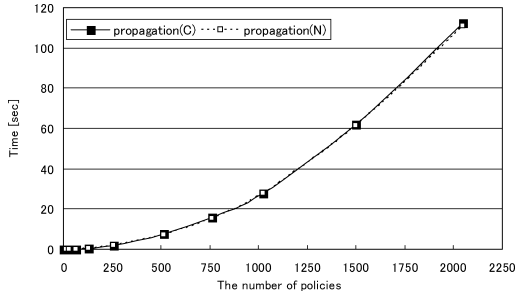

Fig. 9 Time needed to detect an implicit modality conflict.

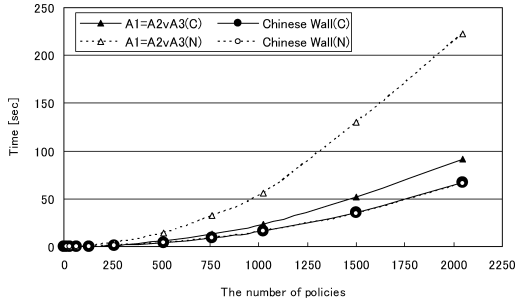

Fig. 10 Time needed to detect a constraint conflict.

because a propagation policy is applied to a role structure and has to be reapplied multiple times in the FVT method to detect a conflict. This becomes increasingly complex as the role structure becomes more complex. As shown in Fig. 10, analyzing the action composition and Chinese wall policies is not as complex as analyzing propagation policies because they do not require recursive analysis.

Finally, **Fig. 11** shows that it took longer to detect a mixed conflict caused by both subject and target propagation policies (indicated as "prop-prop" in Fig. 11) or both propagation and Chinese wall policies (indicated as "prop-CW" in Fig. 11) than to detect a single type of conflict.

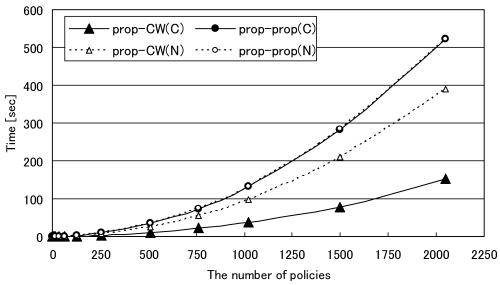

Fig. 11 Time needed to detect a mixed conflict.

The results show that our method completes the analysis even when a set of policies has no conflicts. In most cases, a set of up to about 1,000 policies can be analyzed within about 100 seconds, which is not excessive. Our method should be used statically to detect conflicts before the system starts operating. Therefore, we think these results show that the FVT method is practical since 1,000 policies are enough to manage large systems. On the other hand, the results show that a fairly long time is needed to analysis over 1000 policies when various kinds of policies are included. We think it will be possible to reduce the computational time by specializing the theorem prover for the policy analysis. This will be done as part of our future work.

In general, it cannot be decided whether a first-order logic formula is provable; i.e., that a tableau is terminated in all cases. However, in our application of policy translation we use no function symbols and the counter-models to policy conflicts are therefore finite. By suitably restricting substitutions to variables, we believe our method can be implemented in a way that enables decidability.

8. Extensions

Here, we briefly explain the extensibility of our method. Once policies are translated into first-order logic, all kinds of conflicting policies and redundant policies can be detected using the same algorithm. This means our method can be applied for any type of policy which can be translated into first-order logic. For example, as described in Sections 4.2 and 4.3, an authorization policy and an obligation policy are translated involving an event. Some of these may be composite events – e.g., $E_1 = E_2 \wedge E_3$ – which result in conflicts as indicated in the following.

$$\begin{aligned} \text{r29} &: \texttt{Obli+}(E_1, S_1, T_1, A_1) \\ \text{r30} &: \texttt{Obli-}(E_2, S_1, T_1, A_1) \\ \text{r31} &: \texttt{Obli-}(E_3, S_1, T_1, A_1) \end{aligned}$$

To detect this conflict by FVT, we should just add the mapping relating the events, $\zeta(E_1 = E_2 \wedge E_3) = E_1 \leftrightarrow E_2 \wedge E_3$. We do not need any further changes to detect the conflict. This shows our method can be flexibly extended to cope with changes to the access control policy model. Strictly speaking, our proposed method can be applied to any policy that can be translated into first-order logic; for example, those including time constraints.

9. Conclusion and Future Work

We have proposed a conflict and redundant policy detection method based on free variable tableaux. This method not only detects conflicts but also provides information helpful for correcting policies. It can also be used to detect redundant rules. We proved that this method can detect conflicts completely by defining several types of conflict such as a constraint conflict caused by action composition. Moreover, we showed that our method can be easily extended to deal with various kinds of policy model.

In the future, we will specialize the theorem prover for the particular kinds of sentence occurring as a result of the translations and will compare the computational time with that of similar approaches. We will also investigate extensions to cope with policy models which have more complex constraints, such as delegation policies, time constraint policies, etc. In particular, our preliminary analysis of dealing with time constraints has shown that incorporating abduction into the method will prove useful.

Acknowledgments We thank Syuichiro Yamamoto for his many helpful comments and suggestions.

References

1) Beckert, B. and Goré, R.: Free variable tableaux for propositional modal logics, *Proc. International Conference on Theorem Proving with Analytic Tableaux and Related Methods* (1997).
2) Beckert, B. and Posegga, J.: leanTAP: Lean Tableau-based Deduction, *Journal of Automated Reasoning*, Vol.15, No.3, pp.339–358 (1995). http://i12www.ira.uka.de/leantap/
3) Bertino, E., Bonatti, P.A. and Ferrari, E.: TRBAC: A temporal role-based access con-

trol model, *ACM Transactions on Information and System Security* (*TISSEC*), Vol.4, No.3, pp.191–233 (Aug. 2001).
4) Bhatti, R., Joshi, J.B.D., Bertino, E. and Ghafoor, A.: Access Control in Dynamic XML-based Web-Services with X-RBAC, *Proc. International Conference on Web Services, CWS '03*, June 23–26, 2003, Las Vegas, Nevada, USA, pp.243–249 (June 2003).
5) Brewer, D.F.C. and Nash, M.J.: The Chinese Wall Security Policy, *Proc. IEEE Symposium on Security and Privacy*, May 1–3, 1989, Oakland, California, USA, pp.206–214 (May 1989).
6) Cholvy, L. and Cuppens, F.: Analyzing Consistency of Security Policies, *SP '97: Proc. 1997 IEEE Symposium on Security and Privacy*, pp.103–112 (1997).
7) Clark, D.D. and Wilson, D.R.: A Comparison of Commercial and Military Computer Security Policies, *Proc. IEEE Symposium on Security and Privacy*, April 27–29, 1987, Oakland, California, USA, pp.184–194 (Apr. 1987).
8) Damianou, N., Dulay, N., Lupu, E. and Sloman, M.: The Ponder Policy Specification Language, *Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks*, Bristol, U.K., pp.18–39, Springer-Verlag, LNCS 1995 (Jan. 2001).
9) Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R. and Chandramouli, R.: Proposed NIST standard for role-based access control, *ACM Transactions on Information and System Security*, Vol.4, No.3, pp.224–274 (Aug. 2001).
10) Fitting, M.: *First Order Logic and Automated Theorem Proving*, second edition, Springer (1996).
11) Graham, A., Radhakrishnan, T. and Grossner, C.: Incremental Validation of Policy-Based Systems, *Proc. 5th IEEE International Workshop on Policies for Distributed Systems and Networks* (*POLICY '04*), June 7–9, 2004, Yorktown Heights, New York, pp.240–249 (June 2004).
12) Hayakawa, A., Hoshikawa, T., Takahashi, S. and Kamanaka, H.: A proposal of On-demand VPN architecture, *Proc. 67th National Convention of IPSJ*, Vol.3, pp.329–330 (Mar. 2005).
13) Jajodia, S., Samarati, P. and Subrahmanian, V.S.: A Logical Language for Expressing Authorizations, *Proc. 1997 IEEE Symposium on Security and Privacy*, May 4–7, 1997, Oakland, California, USA, pp.31–42 (1997).
14) Kalam, A.A.El, Benferhat, S., Miége, A., Baida, R.El, Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y. and Trouessin, G.: Organization based access control, *Proc. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, June 4–6, 2003, Lake Como, Italy, pp.120–131 (June 2003).
15) Kalman, J.A. and Wos, L.: *Automated Reasoning with Otter*, Rinton Press (2001).
16) Letz, R. and Stenz, G.: Model elimination and connection tableau procedures, *Handbook of automated reasoning*, pp.2015–2112 (2001).
17) Letz, R. and Stenz, G.: Model elimination and connection tableau procedures, pp.2015–2112 (2001). http://www4.informatik.tu-muenchen.de/~letz/setheo/
18) Lupu, E.C. and Sloman, M.: Conflicts in Policy-Based Distributed Systems Management, *IEEE Transactions on Software Engineering*, Vol.25, No.6, pp.852–869 (Nov. 1999).
19) Massacci, F.: Reasoning about Security: A Logic and a Decision Method for Role-Based Access Control, *Proc. International Joint Conference on Qualitative and Quantitative Practical Reasoning* (*ECSQARU/FAPR '97*), Vol.1244, pp.421–435 (1997).
20) McCune, W.: *OTTER 3.3 Reference Manual and Guide*, Argonne National Laboratory, Illinois (2003). http://www-unix.mcs.anl.gov/AR/
21) OASIS: eXtensible Access Control Markup Language (XACML) Version 2.0, *OASIS Standard* (Feb. 2005). http://www.oasis-open.org/
22) Ong, K. and Lee, R.M.: A decision support system for bureaucratic policy administration: An abductive logic programming approach, *Decision Support Systems*, Vol.16, No.1, pp.21–38 (Jan. 1996).
23) Otten, J. and Bibel, W.: leanCoP: Lean Connection-Based Theorem Proving, *Journal of Symbolic Computation*, Vol.36, pp.139–161 (2003).
24) Ribeiro, C., Zúquete, A., Ferreira, P. and Guedes, P.: Security Policy Consistency, *Technical Report, INESC* (June 2000).
25) Sandhu, R.S.: The Typed Access Matrix Model, *SP '92: Proc. 1992 IEEE Symposium on Security and Privacy*, pp.122–136 (1992).
26) Stickel, M.E.: A prolog technology theorem prover: A new exposition and implementation in prolog, *Technical Note 464, Artificial Intelligence Center, SRI International*, Menlo Park, California (1989).
27) Strembeck, M.: Conflict Checking of Separation of Duty Constraints in RBAC — Implementation Experiences, *Proc. Conference on Software Engineering* (*SE2004*), Innsbruck, Austria, pp.224–229 (Feb. 2004).
28) Sutcliffe, G. and Suttner, C.: The TPTP Problem Library for Automated Theorem Proving. http://www.cs.miami.edu/~tptp/
29) Takeuchi, Y., Hayakawa, A. and Takahashi,

S.: A study of Policy control on On-demand VPN, *Proc. 67th National Convention of IPSJ*, Vol.3, pp.331–332 (Mar. 2005).

(Received May 16, 2005)
(Accepted February 1, 2006)
(Released May 10, 2006)

Hiroaki Kamoda received his M.S. degree from Chiba University in 2000. He has been working in NTT DATA Corp. since 2000. From 2003 to 2004 he was a visiting researcher at FOKUS, Germany, and from 2004 to 2005 he was a visiting researcher at Imperial College London, U.K. He has been engaged in research in the areas of network security and software engineering. He is a member of IPSJ.

Masaki Yamaoka received his M.E. and Ph.D. degrees from Osaka University in 1991 and 2000, respectively. He was a visiting scientist at MIT from 1992 to 1993 and a visiting researcher in the Department of Computing, Imperial College London from 2004 to 2005. His research interests include pattern understanding and network security. He is a member of IPSJ and IEEE.

Shigeyuki Matsuda received his M.S. degree from the University of Electro-Communications in 1981. He worked in the NTT Laboratory from 1981 to 1988 and then at NTT Data from 1988 to 2005. Since 2005, he has been with ERNST & YOUNG SHINNIHON. He is a member of PMI.

Krysia Broda is a Senior Lecturer and member of the Computational Logic section in the Department of Computing, Imperial College London. Her research interests include automated reasoning in classical and non-classical logic, logical agents, and various applications of logic programming including Inductive Logic Programming. She is co-author of several books and has publications in all her areas of interest. She has been a member of various TABLEAUX program committees. See http://www.doc.ic.ac.uk/˜kb for more details and selected papers.

Morris Sloman is Director of Research and Deputy Head of Department in the Department of Computing, Imperial College London. His research interests include management of networks and distributed systems, adaptive security management, trust and security for pervasive systems, and autonomic management for pervasive systems. He has many journal and conference publications and is editor of a reference book on the management of network and distributed systems (published by Addison Wesley). He is a member of the editorial boards of the Journal of Network and Systems Management and the IEEE eTNSM Journal. He is on the steering committees for the conferences on Policies for Distributed Systems and Networks, Integrated Management (IM), and Network Operations and Management (NOMS). See http://www.doc.ic.ac.uk/˜mss for more details and selected papers.