Formal Verification of Security Properties in Trust Management Policy *

Jianwei Niu William H. Winsborough Mark Reith†
University of Texas at San Antonio
One UTSA Circle, San Antonio, Texas, USA 78249
niu@cs.utsa.edu, wwinsborough@acm.org, mark.reith@afit.edu

Abstract

Trust management is a scalable form of access control that relies heavily on delegation. Different parts of the policy are under the control of different principals in the system. While these two characteristics may be necessary in large or decentralized systems, they make it difficult to anticipate how policy changes made by others will affect whether ones own security objectives are met and will continue to be met in the future. Automated analysis tools are needed for assessing this question. The article develops techniques that support the development of tools that nevertheless are able to solve many analysis problem instances. When an access control policy fails to satisfy desired security objectives, the tools provide information about how and why the failure occurs. Such information can assist policy authors design appropriate policies. The approach to performing the analyses is based on model checking. To assist in making the approach effective, a collection of reduction techniques is introduced. We prove the correctness of these reductions and empirically evaluate their effectiveness. While the class of analysis problem instances we examine is generally intractable, we find that our reduction techniques are often able to reduce some problem instances into a form that can be automatically verified.

1 Introduction

Correctly configuring authorization systems is difficult. Effectively and appropriately controlling who has access to which resources has serious implications for the interests of private citizens, corporations and other organizations, including national governments. If the authorization state were static and never needed to be changed, the problem would be less difficult. Unfortunately, it is necessary to maintain high-level security objectives while the authorization state is changing. In one of the classical results [14] in computer security, Harrison, Ruzzo, and Ullman showed in 1976 that a relatively simple, highly desirable security analysis problem is undecidable.

The problem they studied is called *safety*. The problem instance is given by the following. An initial authorization state is specified. In the model they use, the authorization state is given by an access control

*This article is an extended version of the conference paper appeared in M. Reith, J. Niu, and W. Winsborough, "Toward Practical Analysis for Trust Management Policy", *4th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 310–321, March 2009. This article was submitted to the Journal of Computer Security (JCS) in March 2011.

†The author is affiliated with Air Force Institute of Technology. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

matrix having rows and columns for subjects and objects, respectively, and a set of rights that the subject has on the object in the corresponding cell. State transitions occur as the result of executing administrative commands. A set of commands is given by the problem instance. These commands have the following form: they take parameters (subjects and objects); they test whether command-specified rights are elements of command-specified matrix cells; if the specified cells contain the specified rights, a sequence of primitive operations is executed. Primitive operations can add and remove subjects (rows) and objects (columns) and they can add or remove rights from cells. The assumption is that all possible command sequences are possible. Finally, the problem instance includes a specified subject, a specified object, and a specified right. The problem question asks whether there is a transition sequence starting from the initial matrix that makes the cell determined by the given subject and the given object contain the given right.

Clearly this is an important question to be able to answer; an important security objective is likely to be that the specified subject never receive the specified right on the specified subject.

In the years following this troubling undecidability result, enormous effort was devoted to finding authorization models that provide adequate expressive power while admitting efficient decision procedures for the version of the safety problem that can be specified within the model. (Representatives of models that succeeded in this goal include the Take-Grant model of Lipton and Snyder [24], and the TAM model by Sandhu [31].) In the end, few of these models received wide acceptance, being viewed as too hard to use in practice, and the goal of designing authorization systems that support efficient safety analysis was supplanted by other goals, such as usability, flexibility, and scalability. Interestingly, authorization systems that have been proposed relatively recently turn out to allow (the corresponding version of) the classical safety problem to be solved efficiently. In particular, this is true of the authorization system we study here[1].

Classical access control systems (*e.g.*, Bell-LaPadula [5], Role-based Access Control [32], and Domain and Type Enforcement [2]) were designed for monolithic information and control systems used by organizations having rather static structure by today's standards. They tend to centralize administrative authority and to limit their expressive power to characterizing the relationship a principal has to a single, implicity authority, which makes them best suited for use within a single organization, making their application to inter-organizational partnerships cumbersome. They also do not easily accommodate rapid reorganization and its accompanying needs for policy flexibility.

One approach to achieving scalability and flexibility is called *Trust Management* [6]. Trust management supports decentralization of authority by enabling interested parties to define authorization policy without the intervention of a security officer or a handful of security-system administrators. Another approach that has become very popular over the last 10 or 15 years, is *Role-Based Access Control* (RBAC) [32], which assigns users to *roles* based on their organizational and functional roles, and assigns permissions to obtain access to roles, rather than directly to individual users. This approach is viewed as greatly enhancing manageability, although like many other security systems, it tends to focus on the relationship between a principal and a single organization.

A relatively recent contribution that blends strengths from both these approaches is *Role-based Trust management* (RT). RT provides a family of authorization policy languages. The framework provides a collection of language features, each providing a distinct expressive power, and the various languages are obtained by combining the various features. (See for instance [15, 19, 20, 25].) From RBAC, RT borrows the ability to characterize principals much more generally than in terms of the resources to which they should have access. From trust management, it borrows features that support decentralization through delegation of authority by stakeholders such as resource owners to knowledgeable parties that may have greater familiarity or expertise with respect to characterizing would-be resource users. This facilitates both

[1] We go on to study the evaluation of much more expressive policy analyses, which are decidable, but generally intractable.

scalability and flexibility. Each role has an associated *owner* that controls which principals are added to the role. By choosing to add the members of roles owned by other principals, the owner delegates authority to those other roles' owners, who in turn have sole control over the principals added to the roles they own.

The simplest member of the RT family is RT_0. Li *et al.* [21] have shown that many security properties, including the classical safety property, can be decided for RT_0 in polynomial time. However, the most expressive and most useful properties, which require role-containment analysis [21], are EXPTIME-complete [36]. Among these properties are notions of availability, mutual exclusion, and a generalization of the safety problem. Consider the following example of the generalized form of safety: Assuming certain parts of the policy do not change, it will always be the case that only employees can access the confidential database, no matter what changes are made in the rest of the policy. Rather than asking whether a particular principal could gain access, as is done by the classical safety problem, it asks whether a whole class of principals—those that are not employees—could gain access. The importance of gaining assurance that such security objectives will continue to be met as the policy evolves is clear.

The fact that role-containment analysis is EXPTIME-complete indicates that many problem instances will remain forever out of reach. However, this does not preclude the possibility that analysis techniques could be developed that would be useful in many cases of interest. Thus, it is natural to wish to determine the limits of such analyses when applied to realistic policies. This article seeks to do just that. In particular, it assesses the extent to which containment queries can be analyzed by using currently available model-checking technology. When using this approach, one transforms a given security-analysis problem instance into a model checking problem instance. Model checking [7, 8] offers general-purpose verification tools that are fully automated. In essence, they check whether all runs of a given finite state machine satisfy some property. When a run is discovered that does not satisfy the property, it is reported as a counterexample. The tool we use is also highly optimized for exploring large search spaces (which typically grow exponentially in the size of the input). One technique that is often used in conjunction with model checking is called reduction. *Reduction* converts one problem instance to another that is designed to be less expensive to solve.

A security-analysis problem instance $\pi = \langle \mathcal{P}, \mathcal{R}, \mathcal{Q} \rangle$ is given by an RT policy $\mathcal{P}$, a structure $\mathcal{R}$ called a restriction rule, and a query $\mathcal{Q}$. A policy $\mathcal{P}$ is a set of *statements* that assign principals to roles. These statements compose the *definition* of the role and each such statement is said to *define* the role, though in general it actually merely contributes to the definition. The *semantics* of a policy is given by an assignment to each role of a set of principals as members of the role[2]. In addition to being a candidate for membership in roles, each principal has the sole right to create, modify, and remove statements that define the roles she owns. Thus, the administrative model is very simple. When such a statement adds (the members of) a role owned by another principal, the principal authoring that statement does not control the statements that define that other role. In this sense, the statement author is delegating authority to assist in defining the membership of her own role. (It can be assumed that membership in certain roles controls access to various resources, however this association of roles to resources is not established by the RT policy itself.) We call a given policy an *authorization state*, a *policy state*, or simply a *state*. A state transition occurs when a principal creates, modifies, or removes a statement defining one of her roles.

In the absence of any assumptions about which roles are changed, any given state is reachable from any other state. Loosely speaking, a restriction rule $\mathcal{R}$ identifies roles defined by statements that are assumed for the purpose of the analysis not to change. (This notion is made precise in the next section.) In this manner, the restriction rule states assumptions about which state transitions can be taken, and therefore which states are reachable under $\mathcal{R}$ from the initial state $\mathcal{P}$.

[2]Note that many policies may induce the same semantics.

A query $\mathcal{Q}$ relates the membership of a role to some other set. While simpler queries are also supported[3], a *role containment query* asserts that the membership of one role is contained (as a subset) in the membership of another role in the semantics of every reachable state. Thus a query may or may not be satisfied by $\mathcal{P}$ under $\mathcal{R}$. In the example problem instance mentioned above, the query asserts that in all reachable states, the principals that are in the role defining who can gain access to a secret database are all members of the role consisting of all company employees.

The question part of the security-analysis problem asks whether the assertion given by the query holds in (the semantics of) each state reachable from $\mathcal{P}$ under $\mathcal{R}$. We present and evaluate empirically several automated reduction techniques that diminish the computational cost of analysis[4]. These techniques operate at two levels. Techniques in the first class transform a security-analysis problem instance into one or more instances of the same problem that together are equivalent to the original, but that are often less expensive to decide. These techniques can be used with any method of deciding analysis problem instances because they transform one problem instance into one or more instances of the *same* problem. Techniques in the second class simplify the model-checking problems generated by our problem transformation. As such, they would not be applicable with all security-analysis methods. We present three reduction techniques in the first class and two in the second class.

An analysis problem evaluation method is said to be *sound* if whenever it replies in the affirmative (*i.e.*, the query holds in all reachable states), the role-role containment does in fact hold in every reachable state. The method is said to be *complete* if when the role-role containment does in fact hold in every reachable state, the evaluation method replies in the affirmative. All of the reduction techniques mentioned thus far are both sound and complete.

Evaluation methods that are sound, but incomplete or that are complete, but not sound are called *semi-decision procedures*. We present one procedure in each of these categories. Because they require less computational effort, we are particularly interested in semi-decision procedures that require the exploration a smaller set of states than must be explored by our sound and complete evaluation methods (full decision procedures). The semi-decision procedures that we present have this property and hence sometimes terminate successfully when our full decision procedures do not. However, semi-decision procedures provide less information than do full decision procedures. When a sound, but incomplete semi-decision procedure reports that the query holds, we know the query is actually satisfied, confirming that the associated security objective is met, as desired. However, like all procedures in this category, in some cases it may fail to confirm that the query is satisfied when in fact it is. On the other hand, a semi-decision procedure that is complete, but unsound may report that the query holds when in fact it does not. The utility of such a semi-decision procedure is less obvious than those in the first; it is that when a complete but unsound procedure reports that there exists a state that falsifies the query, this is in fact the case. Hence, such a procedure can be useful as a policy "debugger." We call a state in which the role-role containment actually does not hold a *counterexample*. The complete but unsound procedure we present has the property that counterexamples that are found are states that are actually reachable in the original analysis problem instance. Because the counterexample that is discovered is reported by the tool, it can help a policy author to understand how the part of the policy controlled by her and the other role owners that she trusts is vulnerable to having security objectives violated through policy changes made to parts of the policy that are under the control of others.

Recall that our reduction techniques can be divided into (1) those that can be applied with any evaluation

[3] A simpler form of query relates the set of members of a given role to a constant set, specified explicitly in the query, that does not depend on the policy state under consideration.

[4] An important observation that we make use of extensively in our reductions is that it is not essential to explore all reachable policy states in order to answer the security-analysis question—it is sufficient to explore a set of policy states that can be shown to induce exactly the set of role membership assignments (semantics) that are induced by the reachable policy states.

technology and (2) those that are specific to the model-checking approach. Among the two semi-decision techniques, we have one in each of these two classes. The sound, but incomplete one can be applied with any evaluation technology. It replaces the restriction rule with one that is less restrictive. The procedure that is complete, but unsound is specific to the model-checking approach.

Contributions

1. We design and prove correctness of:
   (a) Three reduction techniques that often reduce the computational effort required to perform role containment analysis. These techniques can be applied with any evaluation technology; they produce one or more potentially much less costly instances of the same policy-analysis problem that are all satisfied if and only if the original problem instance is satisfied.
   (b) Techniques for transforming security analysis problem instances into model checking problem instances.
   (c) Two reduction techniques that are specific to our model-checking approach and that often substantially reduce the size of the model-state space to be explored.
   (d) Two semi-decision procedures for policy analysis. Again, these often dramatically reduce the size of either the policy-state space or the model-state space that must be considered. One of these semi-decision procedures is sound, but incomplete and transforms one policy-analysis problem instance into another. The other is complete, but not sound and transforms one model-checking problem instance into another. By using each of these techniques, one can identify many role containments that definitely are satisfied or definitely are not. Both techniques allow the user to adjust the values of certain parameters in a way that yields a trade off between precision and cost.
2. We implement a tool suite called RT-SPACE for performing role containment analysis that implements the designs listed above. The translation tool takes a role containment problem instance and generates a model for input to the Cadence SMV model checker [1, 27]. The model checker provides a counterexample when the problem instance is not satisfied. This facilitates policy "debugging." The suite provides an integrated environment in which to specify and visualize RT policies and role-containment problem instances, as well as to evaluate those problem instances and to visualize their results (*e.g.*, counterexamples).
3. We use RT-SPACE to empirically evaluate the performance of our techniques and to assess the results. To our knowledge, RT policies used in practice are not available. Therefore, by using a combination of manual and automated means, we generate policies for analyzer evaluation that are as realistic as possible. Our findings from this assessment include the identification of features of analysis problem instances that have a large effect on the cost of performing the analysis.

Preliminary reports of this investigation have appeared previously [28, 29]. Let us now summarize the material that has not previously appeared and is new in this archival article. Some reduction techniques have previously been specified or discussed informally [29]. 1(a) Among the three analysis problem-level reduction techniques that we present in the new submission, only one has be previously discussed in its current form. The second has not been previously discussed. The third is strictly more general than the combination of two reduction techniques previously reported and replaces the previous two in the new submission. All of the reduction techniques presented here are shown to be sound and complete; no such

verification has previously appeared. 1(b) The transformation of the security problem into a model checking problem has previously been discussed informally, but has not been precisely specified or verified, both of which it is in the new article. 1(c) The reduction techniques that work at the model-checking level are both new (as are their verifications). 1(d) The semi-decision procedures have previously been discussed informally, but have not been precisely specified or verified. 2 The previously presented [28] tool suite, RT-SPACE, has now been updated to include implementations of the new reduction techniques listed above. 3 The set of features of analysis problem instances that can dramatically influence the cost of policy analysis is new in this article. Guided by these features, the new submission introduces criteria for systematically generating test cases and evaluating the analysis techniques.

The structure of this paper is as follows. Section 2 describes the RT language, role containment analysis, and prior results identifying tractable classes of analysis problem instances. Section 3 describes sound and complete reductions that facilitate our verification techniques successfully terminating. Section 4 presents the construction of finite state machine models from analysis problem instances, which enables the use of model checking for our purposes, as described in section 5. Section 6 discusses two semi-decision procedures, which significantly reduce the size of the search space that must be explored by the model checker. Section 7 presents the implementation of the policy analysis framework RT-SPACE, providing us with the means to evaluate our techniques in section 8. Section 9 compares our framework with related work, and we conclude in Section 10.

2 The RT Policy Language

RT is a family of role-based trust management languages designed to support highly decentralized, attribute-based access control [20]. It enables resource providers to make authorization decisions about resource requesters of whom they have no prior knowledge. This is achieved by delegating authority for characterizing principals in the system to other entities that are in a better position to provide the characterization. For instance, to grant discounted service to students, an electronic bookstore might delegate to universities the authority to identify students and delegate to accrediting boards the authority to identify universities.

A significant problem that policy authors face in this context is that of determining the extent of their exposure through delegation to principals not under their direct control or which they are prepared to trust only partially. The security analysis problem in this context consists of determining whether changes made by such principals could cause certain policy objectives to become violated. One example of the security analysis problem would ask whether changes made by principals anyone outside the electronic publishing organization could cause inappropriate resource requesters to receive the student discount or, more generally, gain access to the organization's sensitive data. This security analysis problem has been studied by Li *et al.* [21] for the simplest member of the RT family, RT_0. While it must be emphasized that there are many languages in the RT family, as we focus exclusively on RT_0, in the interest of brevity, the remainder of this article uses simply RT to refer to RT_0. In this section, we summarize RT and the security analysis of it.

2.1 Overview of RT Syntax & Semantics

The RT language provides two primary constructs, principals and roles. A principal is an entity such as a person or software agent. A role takes the form $A.r$, in which A is a principal and r is a role name. In RT_0, A is the *owner* of $A.r$. One interpretation of the role $A.r$ is that the principal A considers the members, which are also principals, to have an attribute, property, or characteristic denoted by the role name. In an RT policy, A can add principals of A's choosing to role $A.r$, which A does by issuing policy statements,

Type	Syntax	Description
Type I	$A.r \leftarrow D$	Simple Member
Type II	$A.r \leftarrow B.r_1$	Simple Inclusion
Type III	$A.r \leftarrow B.r_1.r_2$	Linking Inclusion
Type IV	$A.r \leftarrow B.r_1 \cap C.r_2$	Intersection Inclusion

Figure 1: RT statements

each of which takes one of the four forms shown by Figure 1 [23]. Each policy statement defines the role on the left-hand side of the arrow and adds principals denoted by the expression the the right-hand side. So each of the statements shown in the figure define the role $A.r$, which means they can be issued only by principal A. Thus, the administrative model is very simple: A and only A can issue or revoke statements that define roles owned by A. Statements are assumed to be verifiable, either cryptographically, or by some other means whereby integrity and authenticity can be ensured.

Refering to Figure 1, a role owner A can add a specific principal D by issuing a type 1 statement. For example, by issuing *Alice.friend* ⟵ *Bob*, Alice identifies Bob as being one of her friends. By issuing a type II statement, A can add all the members of another role, $B.r_1$, as members of $A.r$. In this case, A delegates authority to B to assist in defining $A.r$. For example, by issuing the statement *Alice.friend* ← *Bob.friend*, Alice states that any friend of Bob is also a friend of Alice. So if Bob issues *Bob.friend* ⟵ *Dave*, Dave becomes a member not only of *Bob.friend*, but also of *Alice.friend*. Here Alice delegates authority to a specific principal (Bob) by name. Type III statements provide a mechanism whereby a role owner A can delegate authority to all members of a given role $B.r_1$. For example, the statement *Alice.friend* ← *Bob.family.friend* says that any friend of a member of Bob's family is also a friend of Alice. So any member of Bob's family can contribute to the membership of *Alice.friend*. We call this *attribute-based* delegation because Alice delegates authority to all members of the designated role, rather than to a specific principal identified[5]. Finally, Type IV statements support intersection: a principal must be a member of two given roles, $B.r_1$ and $C.r_2$ in order to be added to $A.r$ according to a statement of this type. For example, *Alice.friend* ← *Bob.friend* ∩ *Carl.friend* says that those principals who are friends of both Bob and Carl are introduced into the set of Alice's friends. Disjunction (set union) is provided by allowing multiple statements to be issued defining the same role. Note that a given principal must appear in the right-hand side of some Type I statement if it is to be contained in any role. We borrow the following example from the literature to illustrate RT to further illustrate the use of the first three types of credentials.

Example 1 ([23]) Suppose EPub considers students of all universities to be entitled to a discount on its publications. If RT did not support attribute-based delegation, EPub would have to know all the universities and delegate explicitly to each of them. Using attribute-based delegation, EPub can delegate authority over identifying universities to another entity, *e.g.*, a fictitious Accrediting Board for Universities, ABU.

EPub.discount ⟵ EPub.university.student	(1)		
StateU.student ⟵ RegistrarB.student	(3)	RegistrarB.student ⟵ Alice	(4)
EPub.university ⟵ ABU.accredited	(5)	ABU.accredited ⟵ StateU	(6)

These credentials form a chain from Alice to EPub.discount, consisting of three parts.

[5]Li *et al.* [23] restrict Type III credentials to have the form $A.r \longleftarrow A.r_1.r_2$. *I.e.*, the principal on the right, B, must be the same as the principal on the left, A. This benefits discovery of proofs of role membership, called *credential chains*. It does not limit expressive power, as A can simply issue the statement $A.r_1 \longleftarrow B.r_1$ to achieve the same effect as the form introduced here. We allow the more general syntactic form here because doing so simplifies many examples.

$$\text{part(a): EPub.discount} \xleftarrow{(1)} \text{EPub.university.student}$$
$$\text{part(b): EPub.university} \xleftarrow{(5)} \text{ABU.accredited} \xleftarrow{(6)} \text{StateU}$$
$$\text{part(c): StateU.student} \xleftarrow{(3)} \text{RegistrarB.student} \xleftarrow{(4)} \text{Alice}$$

Here, part(b) shows that StateU has the attribute on the basis of which EPub delegates to StateU the authority to identify students. In this way, it connects part(a) and part(c) into a chain. Thus the example illustrates how attribute-based delegation promises flexibility and scalability. (Admittedly, the example also illustrates the fact that attribute-based delegation requires that all universities use the same role name to define their student roles, which requires cooperation among all universities if their students are to receive the discount.)

Given a set of RT policy statements, each statement is of the form $A.r \longleftarrow \lambda$ in which $\lambda \in \{D,\ B.r,\ B.r_1.r_2,\ B.r_1 \cap C.r_2\}$. We call the left hand side of the arrow, $A.r$, the *defined role*. For any λ of the form D, $B.r$ $B.r_1.r_2$, or $B.r_1 \cap C.r_2$, we call λ a *role expression*. Given a role expression of the form $B.r_1.r_2$, $B.r_1.r_2$ is called a *linked role expression*, $B.r_1$ is the *base-linked role*, and each role of the form $X.r_2$ is called a *sub-linked role* if the given set of policy statements makes X a member of $B.r_1$.

An RT *policy* or *policy state* $\mathcal{P}$ is a set of RT policy statements. We denote the set of role names occurring in $\mathcal{P}$ by $\mathsf{Names}(\mathcal{P})$; we denote the set of principals in $\mathcal{P}$ by $\mathsf{Principals}(\mathcal{P})$; we denote the set of roles of $\mathcal{P}$ by $\mathsf{Roles}(\mathcal{P}) = \{A.r \mid A \in \mathsf{Principals}(\mathcal{P}) \wedge r \in \mathsf{Names}(\mathcal{P})\}$. Note that roles in $\mathsf{Roles}(\mathcal{P})$ do not necessarily appear in $\mathcal{P}$, but are constructed from principals and role names that do. We denote the *restriction of a policy* $\mathcal{P}$ to a given set of roles τ by $\mathcal{P}\restriction_\tau = \{A.r \leftarrow \lambda \in \mathcal{P} \mid A.r \in \tau\}$.

The semantics of RT can be constructed in many equivalent ways. One of these is to transform a policy into a DATALOG program, which has a well defined semantics that can be computed as a fixpoint of an operator called T_P in which P is the program. For our purposes, our proofs are more concise if we skip the transformation to DATALOG and simply construct the semantics of $\mathcal{P}$ directly as a fixpoint of a variant of this operator, $T_\mathcal{P}$, in which $\mathcal{P}$ is an RT policy.

We define the semantics of RT policy $\mathcal{P}$ by a function that takes a role and returns a set of principals (the role's members). It is denoted by $[\![\cdot]\!]_\mathcal{P} : \mathcal{I}(\mathcal{P})$, in which $\mathcal{I}(\mathcal{P}) = \mathsf{Roles}(\mathcal{P}) \to \wp(\mathsf{Principals}(\mathcal{P}))$ and $\wp$ denotes powerset. So for instance $[\![B.r_1]\!]_\mathcal{P}$ yields the set of principals in role $B.r_1$ under policy $\mathcal{P}$. The set of functions of type $\mathcal{I}(\mathcal{P})$ can be ordered by $\eta_1 \sqsubseteq \eta_2$ *iff* $\eta_1(A.r) \subseteq \eta_2(A.r)$ for all $A.r \in \mathsf{Roles}(\mathcal{P})$ and $\eta_1, \eta_2 : \mathcal{I}(\mathcal{P})$. It is easy to see that $\langle \mathcal{I}(\mathcal{P}), \sqsubseteq \rangle$ forms a complete lattice. The function $[\![\cdot]\!]_\mathcal{P}$ can now be constructed by taking the least fixpoint of a monotonic function over functions, $T_\mathcal{P} : \mathcal{I}(\mathcal{P}) \to \mathcal{I}(\mathcal{P})$.

Definition 1 (Semantics of RT) *The $T_\mathcal{P}$ operator is defined as follows. Given any $\eta \in \mathcal{I}(\mathcal{P})$ and any $A.r \in \mathsf{Roles}(\mathcal{P})$,*

$$\begin{aligned} T_\mathcal{P}(\eta)[\![A.r]\!] = \{D \mid\ & A.r \leftarrow D \in \mathcal{P} \vee \\ & (A.r \leftarrow B.r_1 \in \mathcal{P} \wedge D \in \eta[\![B.r_1]\!]) \vee \\ & (A.r \leftarrow B.r_1.r_2 \in \mathcal{P} \wedge \exists Z.Z \in \eta[\![B.r_1]\!] \wedge D \in \eta[\![Z.r_2]\!]) \vee \\ & (A.r \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P} \wedge D \in \eta[\![B.r_1]\!] \wedge D \in \eta[\![C.r_2]\!])\} \end{aligned}$$

The least fixpoint of $T_\mathcal{P}$ can be computed by constructing the Kleene sequence as follows:

$$T_\mathcal{P} \uparrow^0 = \eta_0, \text{ in which } \eta_0[\![A.r]\!] = \emptyset \text{ for all } A.r \in \mathsf{Roles}(\mathcal{P})$$
$$T_\mathcal{P} \uparrow^{i+1} = T_\mathcal{P}(T_\mathcal{P} \uparrow^i)$$

The limit of this sequence, $T_\mathcal{P} \uparrow^\omega$, is given by $T_\mathcal{P} \uparrow^\omega [\![A.r]\!] = \bigcup_{i < \omega} T_\mathcal{P} \uparrow^i [\![A.r]\!]$, for all $A.r \in \mathsf{Roles}(\mathcal{P})$. Because $T_\mathcal{P}$ is monotonic, $T_\mathcal{P} \uparrow^i$ is an increasing sequence and because the number of principals and role

names in $\mathcal{P}$ is finite, $T_{\mathcal{P}} \uparrow^i$ converges at some finite stage i. We now define the semantics as the function $[\![\cdot]\!]_{\mathcal{P}}$ that given any role $B.r_1$, is defined by $[\![B.r_1]\!]_{\mathcal{P}} = T_{\mathcal{P}} \uparrow^{\omega} [\![B.r_1]\!]$.

2.2 RT Policy Analysis

The policy analysis we study [21] determines whether a specified subset relationship between roles holds in all reachable states. We make the notion of reachability precise below; loosely speaking a policy state is reachable from $\mathcal{P}$ if it can be obtained from $\mathcal{P}$ by changing policy statements defining roles that are not explicitly assumed to remain unchanged.

A *query* asks whether a set containments holds in all reachable states $\mathcal{P}'$. Because the containment must hold in *all* reachable states, we use a notation for queries that is reminicent of but distinct from subset notation: $\varrho \sqsupseteq \lambda$ in which each of ϱ and λ is either a role or an explicit (constant) sets of principals. For instance, $X.u \sqsupseteq A.r$ holds if each member of $A.r$ is a member of $X.u$ in every policy state $\mathcal{P}'$ reachable from $\mathcal{P}$, *i.e.*, $[\![A.r]\!]_{\mathcal{P}'} \supseteq [\![X.u]\!]_{\mathcal{P}'}$. Queries of this form can be used to express many important kinds of security properties, including availability, safety, liveness and mutual exclusion. These queries can be used to express security objectives of the entity running the analysis. For instance, one important safety property might be verified by posing the query $\mathrm{employee} \sqsupseteq \mathrm{accessSecretDB}$. The answer "yes" means that no one outside the company has access to the secret database in any reachable state. Note that containment queries such as this are strictly more powerful that the classical safety problem, which asks whether a particular individual, say Alice, is a member of a given role, say $B.r_1$, in any reachable state. This problem can be encoded by adding two new auxiliary roles $C.r_2$ and $D.r_3$, each defined by a single statement, $C.r_2 \longleftarrow Alice$ and $D.r_3 \longleftarrow B.r_1 \cap C.r_2$, and posing the query $\{\} \sqsupseteq D.r_3$.

Reachability is defined precisely as follows. A *restriction rule* is given by a pair of sets of roles $\mathcal{R} = (\mathcal{G}_{\mathcal{R}}, \mathcal{S}_{\mathcal{R}})$. Roles in $\mathcal{G}_{\mathcal{R}}$ are said to be *growth-restricted*, and it is assumed that these roles will have no new statements defining them added in a state transition. Shrink-restricted roles ($\mathcal{S}_{\mathcal{R}}$) are assumed not to have statements defining them removed.

Definition 2 *We say that $\mathcal{P}'$ is* reachable *from $\mathcal{P}$* under *$\mathcal{R}$ if $\mathcal{P}' \restriction_{\mathcal{G}_{\mathcal{R}}} \subseteq \mathcal{P}$ and $\mathcal{P} \restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}'$. In this case we write $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$.*

We can now define the role-containment problem.

Definition 3 (RCPI) *A* role-containment problem instance *(RCPI) is given by a triple $\pi = \langle \mathcal{P}, \mathcal{R}, \mathcal{Q} \rangle$, in which $\mathcal{P}$ is a policy, $\mathcal{R}$ is a restriction rule, and for some $X.u, A.r \in Roles(\mathcal{P})$, $\mathcal{Q} = X.u \sqsupseteq A.r$. The RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is said to be* satisfied *if and only if $[\![X.u]\!]_{\mathcal{P}'} \supseteq [\![A.r]\!]_{\mathcal{P}'}$ for each $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$. In this case we also say that $\mathcal{P}$* satisfies *$X.u \sqsupseteq A.r$* under *$\mathcal{R}$. The question part of the role-containment problem asks whether this is the case.*

One usage of the role-containment problem is in the maintenance of security objectives as uncooperative principals modify the definitions of the roles they own. The presumption is that the owners of shrink- and growth-restricted roles cooperate in maintaining the security objectives by not committing policy changes that violate $\mathcal{R}$ without first running the analysis. This analysis consists of posing the queries that verify security objectives will continue to be met as the definitions of other roles are changed.

Note that the assumptions expressed by the restriction rule are not enforced by an administrative policy. They are simply assumptions under which the analysis is performed. Intuitively, their presence enables the analysis to provide us with assurance of statements such as, "So long as people I trust do not make certain

changes to the definitions of certain roles under their control without first verifying the security objective given by the query $\mathrm{employee} \sqsupseteq \mathrm{accessSecretDB}$, only company employees will be able to access the secret database."

As noted in the introduction, queries of certain restricted forms can be analyzed and verified efficiently, *i.e.*, in polynomial time. These include queries in which at least one of ϱ and λ is an explicit, constant set of principals given in the query [21]. Thus the classical safety problem can be answered in polynomial time in RT. Queries can also be answered in polynomial time if policy states are restricted to contain only Type I and Type II statements. However, when both ϱ and λ are roles and all forms of statements are allowed, the decision problem is EXPTIME-complete [36]. This is unfortunate because the properties that can be expressed by using such queries are extremely useful, as illustrated by the secret database example we have discussed just above and in the introduction. Thus we seek techniques that can solve queries of this general form as often as possible and assess the limits of the techniques we design.

2.3 Prior State Space Reductions

We make extensive use of two prior results that limit the space of reachable states that must be explored to yield a complete RCPI decision. The first justifies adding only simple member statements when constructing reachable policy states.

Theorem 4 ([21]) *Given an RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, if $X.u$ does not contain $A.r$, then there exists a $\mathcal{P}'$ reachable from $\mathcal{P}$ and principal E such that $E \in [\![A.r]\!]_{\mathcal{P}'}$, $E \notin [\![X.u]\!]_{\mathcal{P}'}$, $\mathcal{P}' - \mathcal{P}$ has only simple member statements, and $\mathcal{P}'$ uses only roles names in $\mathcal{P}$.*

The second prior result limits the size of the policy state space that must be explored to determine whether an RCPI is satisfied. This result established an upper bound on the complexity of the role containment problem, though Sistla and Zhou [36, 37] later showed a tighter, precise upper bound.

Any analysis technique that operates by exhaustively examining reachable states must address the fact that in our analysis problem the size of reachable policy states is unbounded, and hence the state space is infinite. The following definition and theorem establish a bound on the size of the states that must be considered.

Definition 5 ($\mathcal{N}(\pi)$) *Given any RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, let $\mathcal{N}(\pi) = \{A.r \longleftarrow D \mid A.r \notin \mathcal{G}_{\mathcal{R}} \wedge r \in \mathsf{Names}(\mathcal{P}) \wedge A, D \in \mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)\}$, in which $\mathsf{NewPrinc}(\pi)$ is a set of new principals, disjoint from $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{Principals}(\mathcal{R})$ and whose cardinality is of size $2^{|\mathsf{SigRoles}(\pi)|}$, and $\mathsf{SigRoles}(\pi)$ is the set $\{X.u\} \cup \{B_1.r_1 \mid B.r \longleftarrow B_1.r_1.r_2 \in \mathcal{P}\} \cup \{B_1.r_1, B_2.r_2 \mid B.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P}\}$.*

Theorem 6 ([21]) *Given any RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, π is not satisfied if and only if there exists a $\mathcal{P}''$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}''$, $\mathcal{P}'' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$ and there exists a principal E such that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$.*

While the theorem as stated here differs from that of Li *et al.* [21], it follows easily as a corollary based on the proof given there[6].

[6]The theorem of Li *et al.* states that the problem we study is in **coNEXP**.

3 Reductions

This section describes several reductions that transform one RCPI into another that is typically less expensive to evaluate. These techniques in many cases enable us to reduce the size of the state space that must be explored. The idea is that given an initial state $\mathcal{P}$, we can often perform the analysis using a smaller initial state and obtain identical results. Our findings in Section 8 indicate that, when using our model checking technique and our platform configuration, these reductions often make the difference between being unable to evaluate an RCPI and being able to do so. Furthermore, we conjecture that they may be applied in general to increase the efficiency of other RCPI-solving approaches, such as one based on the proof method of Sistla and Zhou [36, 37].

3.1 Cone of Influence Reduction

A given RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ may contain statements in $\mathcal{P}$ that are unnecessary for determining the satisfiability of an RCPI. Such extraneous statements can safely be removed in order to reduce the complexity of the problem. This reduction removes statements that are said to be outside of the queries' cone of influence (COI), but goes well beyond the cone of influence commonly associated with model checking [8]. Our version is tailored to role containment policy analysis, but it is not so specific that it cannot be leveraged by other analysis techniques outside of model checking. We describe the specific differences at the end of this section.

Our COI reduction accomplishes three goals. First, it removes any statement that does not contribute to the membership of the queried roles. Second, it modifies $\mathcal{R}$ to constrain the reachable policy state space in a safe manner. Finally, it omits certain statements whose influence on a queried role is accounted for by $\mathcal{R}$. Thus our reduction takes as input an RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ and constructs a new RCPI $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A.r \rangle$ that produces the same answer. The significance of this reduction is that it has the potential to remove linked role or intersection inclusion type statements that enlarge the number of principals necessary to satisfy the RCPI, and in doing so increases the size of the model checking state space. It is also significant because this reduction is not specific to our model checking analysis approach, but rather can be leveraged by other analysis techniques. We begin by describing COI with a high level description and an example, followed by the formal definition and proof of correctness.

Speaking informally, we use the terms *influence* or *depends on* to express the idea that in order to determine the membership of one role λ in any given state $\mathcal{P}$, you would need to consider some $\mathcal{P}' \subseteq \mathcal{P}$. We say only these statements $\mathcal{P}'$, and the roles they define, influence the membership of λ, or that λ depends only on these statements $\mathcal{P}'$. It is not required that a statement actually introduce any principal into λ to be influential, only that such a statement defines λ directly or indirectly through other roles.

The COI ($\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$) retains from $\mathcal{P}$ those statements that influence the memberships of the queried roles. Recall that role containment analysis involves the exploration of states reachable from $\mathcal{P}$, where we seek to find some counterexample state $\mathcal{P}''$ and E such that $E \in [\![A.r]\!]_{\mathcal{P}''} - [\![X.u]\!]_{\mathcal{P}''}$. Observe from this definition that we can assist in this exploration of policy states by reducing the membership of $X.u$ and enlarging the membership of $A.r$. This can be achieved by constructing the union of two sets of statements from $\mathcal{P}$. The first set of statements minimizes the membership of $X.u$ by including only those statements that influence $X.u$ and define shrink restricted roles. This set excludes unnecessary and removable statements to reduce the membership of $X.u$. The second set of statements attempts to enlarge the membership of $A.r$ by including only those statements that influence $A.r$ and define growth restricted roles. Observe that such a statement may define a growth restricted role so that it depends upon non-growth restricted roles, thus providing a means of enlarging the membership of $A.r$. Any statement that influences

$A.r$ but does not define a growth restricted role can be safely excluded because it cannot introduce anything into $A.r$ that could not be introduced directly. Taking the union of these two sets of statements ensures that every statement that influences either $X.u$ or $A.r$ is included in the reduced policy $\mathcal{P}'$.

Our COI reduction produces not only a reduced $\mathcal{P}'$, but it also constructs $\mathcal{R}'$ by adding roles to $\mathcal{R}$ in such a way as to safely reduce the number of reachable policy states we would need to examine under analysis. We calculate the set of roles Γ that contribute only to the membership of $X.u$ and not $A.r$, and define $\mathcal{G}'_{\mathcal{R}} = \mathcal{G}_{\mathcal{R}} \cup \Gamma$. Clearly such a change will only affect the membership of $X.u$ and ensure that we avoid examining policy states where E happens to be in $X.u$, but is not forced to be in $X.u$. Recall from our description of a counterexample state $\mathcal{P}''$ that $E \notin [\![X.u]\!]_{\mathcal{P}''}$. This change in $\mathcal{R}$ assists the search for $\mathcal{P}''$ by preventing any principal that is not forced to be in $X.u$ from being introduced unless it was introduced through some means other than the roles of Γ. Furthermore, we calculate the set of roles Σ that contributes only to the membership of $A.r$ and not $X.u$, and define $\mathcal{S}'_{\mathcal{R}} = \mathcal{S}_{\mathcal{R}} \cup \Sigma$. As with the previous case, it is clear that such a change will only affect the membership of $A.r$, and that we avoid examining policy states where E happens to be absent from $A.r$. This change in $\mathcal{R}$ assists in the search for $\mathcal{P}''$ by preventing E from being removed in the case that it was introduced through Σ.

To illustrate, consider the example in Figure 2 where RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is the initial problem, and $COI(\pi)$ is a new RCPI constructed from the application of the Cone of Influence Reduction. Let us begin by comparing $\mathcal{P}$ and $\mathcal{P}'$ from COI (π). We see that this reduction removes statements 7 through 18, even though many of these statements define roles that are growth or shrink restricted. When comparing the role dependency graph[7] of Figures 2a and 2b, it is not difficult to understand that statements 16 through 18 were removed because the memberships of $X.u$ and $A.r$ do not depend on the memberships of the roles defined by these statements. What is less obvious is the removal of statements 7 through 15 since the existence of these statements does influence the memberships of the queried roles. We examine this part of the reduction with respect to $X.u$ and $A.r$. The reduction with respect to $X.u$ removes statements 7 through 12, while the reduction with respect to $A.r$ removes statements 7 and 13 through 15.

To understand why the reduction with respect to $X.u$ is safe, recall that we desire to find a set of policy states that minimize the membership of $X.u$ to include only those principals that are forced to be in this role. Principals not forced to be in $X.u$ do not aid in the discovery of a reachable $\mathcal{P}''$ and E such that $E \in [\![A.r]\!]_{\mathcal{P}''} - [\![X.u]\!]_{\mathcal{P}''}$. From the role $X.u$ in Figure 2a, traverse the graph only along bold edges representing statements that are forced to be in every policy state due to $\mathcal{S}_{\mathcal{R}}$. The set of statements corresponding to the traversed edges produces the minimum membership of $X.u$. Thus statements 0 through 3 would be in $\mathcal{P}'$, but statements 7 through 11 would not. Furthermore, with the removal of statement 9, $E.r$ no longer influences $X.u$, and thus statement 12 may also be removed despite the fact that $E.r \in \mathcal{S}_{\mathcal{R}}$.

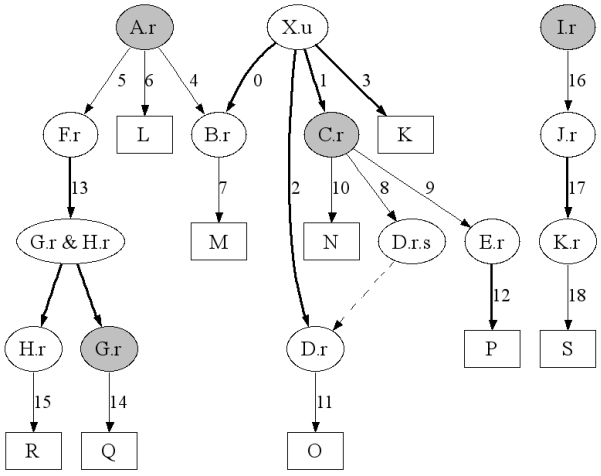
Now not only do we seek to remove such statements, but we also desire to change $\mathcal{R}$ in a way that prevents new statements from introducing principals into $X.u$. We accomplish this by defining Γ as a set of roles that influence $X.u$ but not $A.r$, where $\mathcal{G}'_{\mathcal{R}} = \mathcal{G}_{\mathcal{R}} \cup \Gamma$. In our example, Γ consists of the set $\{X.u, C.r, D.r, E.r\}$. Clearly evaluating policy states that introduce additional principals into $X.u$ but not

[7] We use role dependency graphs (RDG) to illustrate the relationships between roles and between roles and principals. An RDG is a directed graph where each oval node represents a *role expression*, *i.e.* a role, a linked role, the intersection of two roles, or a principal. A role is shaded if it is growth restricted. Each rectangle node represents a principal. Each directed edge represents a dependency of one role expression on another. A solid edge represents a policy statement, labeled with the index of the statement. An edge is in bold signifies that the statement it represents cannot be removed due to the shrink restriction. A dashed edge represents the dependency of a linked role on its base-linked role. Intersection role expressions are depicted using & instead of $\cap$. Two edges originating at an intersection node represent the decomposition of an intersection expression; the destination nodes are the two intersected roles. These edges do not represent policy statements, but are included simply to identify the relationship between an intersection expression and its components.

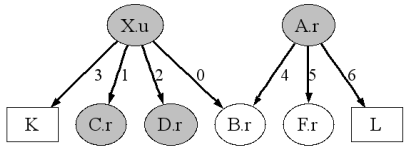
Index	Statement						
0	$X.u \leftarrow B.r$	5	$A.r \leftarrow F.r$	10	$C.r \leftarrow N$	15	$H.r \leftarrow R$
1	$X.u \leftarrow C.r$	6	$A.r \leftarrow L$	11	$D.r \leftarrow O$	16	$I.r \leftarrow J.r$
2	$X.u \leftarrow D.r$	7	$B.r \leftarrow M$	12	$E.r \leftarrow P$	17	$J.r \leftarrow R$
3	$X.u \leftarrow K$	8	$C.r \leftarrow D.r.s$	13	$F.r \leftarrow G.r \cap H.r$	18	$K.r \leftarrow S$
4	$A.r \leftarrow B.r$	9	$C.r \leftarrow E.r$	14	$G.r \leftarrow Q$		

$\mathcal{G}_\mathcal{R} = \{A.r, C.r, G.r, I.r\}$, $\mathcal{S}_\mathcal{R} = \{E.r, F.r, J.r, X.u\}$, $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$

Policy State	Statement Indices
$\mathcal{P}$	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
$COI(\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle)$	0, 1, 2, 3, 4, 5, 6



a. $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$



b. $COI(\pi) = \langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A.r \rangle$, $\mathcal{R}' = (\mathcal{G}'_\mathcal{R}, \mathcal{S}'_\mathcal{R})$, $\mathcal{G}'_\mathcal{R} = \mathcal{G}_\mathcal{R} \cup \{X.u, D.r\}$, $\mathcal{S}'_\mathcal{R} = \mathcal{S}_\mathcal{R} \cup \{A.r, F.r\}$

Figure 2: Cone of influence example

$A.r$ is unnecessary because if E exists in a counterexample state $\mathcal{P}''$, then E must not be introduced into $X.u$ through one of these statements. Thus we need not consider states that contain additional principals in $X.u$ but not $A.r$.

To understand why the reduction with respect to $A.r$ is safe, recall that we seek to increase the membership of $A.r$ as much as possible in order to quickly identify a counterexample should one exist. More precisely, we seek to increase the membership of $A.r$ without influencing the membership of $X.u$ (recall that we simultaneously attempt to minimize the membership of $X.u$). Thus we desire to modify the RCPI to force as many sets of principals into $A.r$ so long as doing so would not affect the membership of $X.u$. One approach involves identifying Σ as a set of roles that influence $A.r$ but not $X.u$, and then adding this set of roles to $\mathcal{S}_\mathcal{R}$. In our example, Σ includes $\{A.r, F.r, H.r, G.r\}$ and we modify $\mathcal{S}_\mathcal{R}$ to create $\mathcal{S}'_\mathcal{R} = \mathcal{S}_\mathcal{R} \cup \Sigma$. In doing so, we ensure that statements 4 through 6 are present in all states investigated. Now if we examine $B.r$ and $F.r$ as roles that influence $A.r$, we see that each of these roles are not growth restricted. In the case of $B.r$, it influences both $X.u$ and $A.r$, so it would not be appropriate to force any particular membership

into $B.r$ as this may influence the membership of $X.u$. In the case of $F.r$, statement 13 can be omitted since the effect it will have on the membership of $A.r$ and $X.u$ will be achieved through statements defining $F.r$ in reachable states. For example, consider any reachable state where $H.r \leftarrow Q$ is added to $\mathcal{P}$, and thus Q becomes a member of $A.r$. The same effect can be accomplished by adding $F.r \leftarrow Q$ to $\mathcal{P}$ where Q is a member of $A.r$, and $X.u$ remains the same. We take the union of the policies produced by both parts of this reduction. In other words, a statement of $\mathcal{P}$ is retained with respect to $\mathcal{R}$ if and only if it influences $X.u$, $A.r$, or both.

The definition of COI makes use of a set, $\mathsf{DefRoles}(\mathcal{P}, \rho, \mathcal{M})$, consisting of roles on which a queried role ρ depends. The definition of this set is parameterized by a set of roles $\mathcal{M}$. It traverses roles in $\mathcal{M}$ on which ρ depends, truncating the traversal paths upon reaching roles not in $\mathcal{M}$. This truncation enables us to define COI aggressively, retaining only those statements on which ρ depends via roles that are growth or shrink restricted. When such truncation is not desired, the value of $\mathcal{M}$ given in the use of DefRoles is the set of all roles in $\mathcal{P}$.

Definition 7 *Let ρ be a role, $\mathcal{M}$ be a set of roles, and $\mathcal{P}$ be a policy. We define $\mathsf{DefRoles}(\mathcal{P}, \rho, \mathcal{M})$ to be the least set of roles $\mathcal{O}$ satisfying the following conditions:*

- $\rho \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1 \in \mathcal{P} \wedge B.r_1 \in \mathcal{M}) \Rightarrow B.r_1 \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1.r_2 \in \mathcal{P} \wedge D \in Principals(\mathcal{P})) \Rightarrow ((B.r_1 \in \mathcal{M} \Rightarrow B.r_1 \in \mathcal{O}) \wedge (D.r_2 \in \mathcal{M} \Rightarrow D.r_2 \in \mathcal{O}))$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P}) \Rightarrow ((B.r_1 \in \mathcal{M} \Rightarrow B.r_1 \in \mathcal{O}) \wedge (C.r_2 \in \mathcal{M} \Rightarrow C.r_2 \in \mathcal{O}))$

Using DefRoles, we define the *Cone of Influence* (COI) as a policy constructed from those statements that define roles in DefRoles. In other words, we retain only those statements that influence the queried roles and thus determine the satisfiability of the RCPI.

Definition 8 *Given an RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, we define $COI(\pi) = \langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A.r \rangle$ in which*

$$\mathcal{P}' = \mathcal{P}\restriction_{\mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R}) \cup \mathsf{DefRoles}(\mathcal{P}, A.r, \mathcal{G}_\mathcal{R})} \quad (1)$$

$$\mathcal{R}' = (\mathcal{G}'_\mathcal{R}, \mathcal{S}'_\mathcal{R}) \quad (2)$$

$$\mathcal{G}'_\mathcal{R} = \mathcal{G}_\mathcal{R} \cup (\mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P})) - \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P}))) \quad (3)$$

$$\mathcal{S}'_\mathcal{R} = \mathcal{S}_\mathcal{R} \cup (\mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P})) - \mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P}))) \quad (4)$$

The function DefRoles traverses roles at most once and for each role determines membership in $\mathcal{R}$. Given a policy of size n roles and $\mathcal{R}$ of size k, the computational complexity is $O(k \log k + n \log k)$ in general and $O(n \log n)$ when the set of roles in $\mathcal{R}$ is a subset of the roles of $\mathcal{P}$, which is typically the case.

Theorem 9 *Given any RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, $COI(\pi)$ is satisfied if and only if π is satisfied.*[8]

Our reduction is distinct from the cone of influence technique expected in model checking. In model checking [8], "the cone of influence reduction attempts to decrease the size of the state transition graph by focusing on the variables of the system that are referred to in the specification." Our version also excludes variables (policy state, role membership) that do not influence the specification, however it goes further

[8]Proof for this and subsequent theorems are given in the Appendix.

in two important ways. First, it recognizes that the removal of linked inclusion and intersection inclusion statements safely reduces the state space by considering fewer principals. Second, it safely reduces the state space by conservatively modifying $\mathcal{R}$. Thus our approach subsumes the common cone of influence technique.

3.2 Empty Role Reduction

The empty role reduction removes policy statements that use in their bodies roles that have no members in any reachable state. While such roles may be uncommon in original RCPIs, the COI reduction can create them. Thus, the two reductions can interact to yield better results than either one alone.

By empty role, we refer to a role whose membership is empty. If we can show that the membership of a given role will be empty in every reachable state, then we can remove statements that reference this role in their body since the statement cannot contribute to the membership of any other role. For example, suppose that $A.r \leftarrow B.r$ is the sole statement defining role $A.r$ and that the $B.r$ is defined by no statements and is growth restricted. In this case, the membership of $B.r$ will be empty in all reachable policy states, so the statement $A.r \leftarrow B.r$ can safely be removed.

The above simple case can be generalized to situations in which the statements defining a collection of growth-restricted roles form a strongly connected component that depends on no outside roles. A simple example of such a policy that contains a single cycle is shown in Figure 3. This policy contains a cyclic dependency created by statements 4, 5 and 6. (Recall that the arc from $C.r\&D.r$ to $C.r$ does not represent a policy statement.) In this case, because the roles that form the cycle are growth restricted and are not defined by any statements that do not participate in the cycle, all these roles, their definitions, and statements that use them can safely be removed from the initial policy. Clearly $B.r$, $D.r$, and $E.r$ have no members in any reachable policy state.

Rather than rely on syntactic means to identify such roles, we make use of a program transformation due to Li *et al.* called the Upper Bound Program and denoted by $UB(\mathcal{P})$ [21]. Li introduces this transformation for the purpose of efficiently evaluating queries in which only the right-hand argument to $\sqsupseteq$ is a role. In this case, it is sufficient to compute a representation of the largest membership that this role can have in a reachable policy state. (Because the membership of growth-unrestricted roles can be unbounded and contain arbitrary principals added in reachable policies, a special pseudo-principal $\top$ is added during the transformation that denotes any (finite) set of principals. Any unrestricted role will have $\top$ among its members, representing the fact that the role's membership can be an arbitrarily large set.)

When the role membership is computed for the transformed program $UB(\mathcal{P})$, any role that is found to be empty will be empty under all reachable policy states. Because of the correctness result for $UB(\mathcal{P})$, these are exactly the roles that the empty role reduction should eliminate from the initial policy. That is, each statement that uses any of these roles in its body should be removed. This is a novel use of $UB(\mathcal{P})$, which was designed for the purpose of answering queries, not transforming initial policies as we are using it here. In the example of Figure 3, the roles $\{B.r, D.r, E.r\}$ would be identified as empty roles under the $UB(\mathcal{P})$ program and thus statements such as 4, 5 and 6 that reference these roles would be removed. The empty role reduction $ERR(\mathcal{P}, \mathcal{R})$ is defined as follows.

Definition 10 *Given a policy $\mathcal{P}$ and restriction rule $\mathcal{R}$, the result of the* empty role reduction *on these values is given by* $ERR(\mathcal{P}, \mathcal{R}) = \{\lambda \leftarrow e \in \mathcal{P} \mid e \in \{B.r, B.r.r_1, B.r \cap C.r_1, C.r_1 \cap B.r\} \wedge [\![B.r]\!]_{UB(\mathcal{P})} \neq \emptyset\}$.

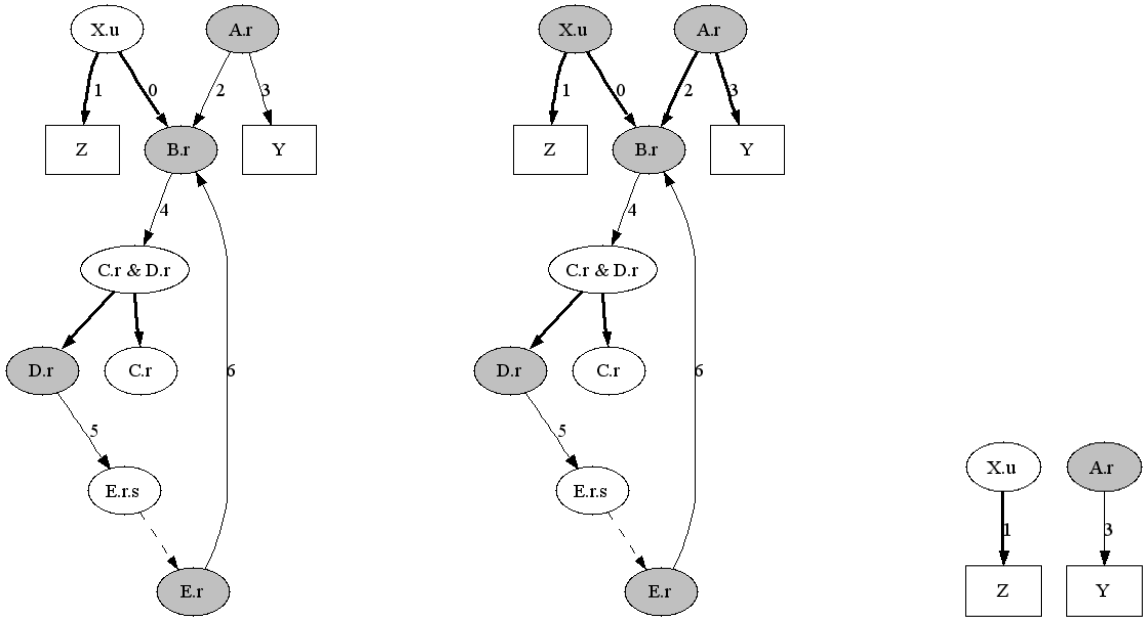
There are two additional points to be made regarding the application reductions to the example of Figure 3. First, the COI reduction does not simplify the initial policy because $B.r$, $D.r$, and $E.r$ each influence

Index	Policy Statement of $\mathcal{P}$		
0	$X.u \leftarrow B.r$	4	$B.r \leftarrow C.r \cap D.r$
1	$X.u \leftarrow Z$	5	$D.r \leftarrow E.r.s$
2	$A.r \leftarrow B.r$	6	$E.r \leftarrow B.r$
3	$A.r \leftarrow Y$		

$\mathcal{G}_\mathcal{R} = \{A.r, B.r, D.r, E.r\}$, $\mathcal{S}_\mathcal{R} = \{X.u\}$

$\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$

Policy State	Statement Indices
$\mathcal{P}$	0, 1, 2, 3, 4, 5, 6
COI ($\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$)	0, 1, 2, 3, 4, 5, 6
ERR ($\mathcal{P}$)	1, 3



a. $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ **b.** $COI(\pi)$ **c.** ERR ($\mathcal{P}$)

Figure 3: Empty role reduction example

the membership of both queried roles. Second, the empty role reduction safely removes statement 0 despite the fact that $X.u$ is shrink restricted. This is possible because $B.r$ is always empty and thus no principals can be introduced through statement 0. We now formally prove the correctness of this reduction.

Theorem 11 *Let $\mathcal{P}$ be any policy and $\mathcal{R}$ be any restriction rule. (1) For all $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$ and $\mathcal{P}' - \mathcal{P}$ contains type 1 statements only, there exists $\mathcal{P}''$ such that $ERR(\mathcal{P}, \mathcal{R}) \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$, $\mathcal{P}'' - ERR(\mathcal{P}, \mathcal{R})$ contains type 1 statements only, and for all $B.r \in \mathsf{Roles}(\mathcal{P}') \cup \mathsf{Roles}(\mathcal{P}'')$, $[\![B.r]\!]_{\mathcal{P}'} = [\![B.r]\!]_{\mathcal{P}''}$. Conversely, it is also the case that (2) for all $\mathcal{P}''$ such that $ERR(\mathcal{P}, \mathcal{R}) \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$ and $\mathcal{P}'' - ERR(\mathcal{P}, \mathcal{R})$ contains type 1 statements only, then there exists $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$, $\mathcal{P}' - \mathcal{P}$ contains type 1 statements only, and for all $B.r \in \mathsf{Roles}(\mathcal{P})$, $[\![B.r]\!]_{\mathcal{P}'} = [\![B.r]\!]_{\mathcal{P}''}$.*

The computational complexity of $UB(\mathcal{P})$ is $O(n^3)$ [21], the construction of a balanced tree of identified empty roles is $O(n \log n)$ since there are at most $O(n)$ roles. Iterating through the policy statements, we incur the cost of $O(n \log n)$ to identify and remove statements that reference empty roles. Thus the computational complexity of this reduction is bounded by $O(n^3)$.

3.3 Decomposition

It is sometimes useful to decompose an RCPI into sub-problems that can be solved separately. The membership of $A.r$ is the union of $[\![e_i]\!]_{\mathcal{P}}$ for $i = 1 \ldots n$ in which $e_1 \ldots e_n$ enumerates $\{e \mid A.r \leftarrow e \in \mathcal{P}\}$. Observe that if $A.r$ is growth and shrink restricted, then for each reachable state $\mathcal{P}'$ and each principal $E \in [\![A.r]\!]_{\mathcal{P}'}$, there is some $i \in \{1 \ldots n\}$ such that $E \in [\![e_i]\!]_{\mathcal{P}'}$. By isolating e_i through the use of a new role $A'.r'$, we construct from a given RCPI a collection of new RCPIs such that the original is satisfied just in case each RCPI in the collection is satisfied. The decomposed RCPIs can sometimes be successfully solved by our analysis tool when the original cannot. We first illustrate this reduction with the following example and then examine the formal definition and proof.

Consider the example in Figure 4 where $\mathcal{P}$ is the initial policy, and RCPI $\pi_1 = \langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq B.r \rangle$, RCPI $\pi_2 = \langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq C.r \rangle$, and RCPI $\pi_3 = \langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A'.r' \rangle$ are three decomposed, sub-problem instances of the role containment problem. A theorem presented below shows that $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is satisfied if and only if these three instances of the role containment problem satisfied. It is interesting to note that the COI reduction is unable to remove any statements in this example because all of the statements influencing $X.u$ define shrink restricted roles, and all of the statements influencing $A.r$, but not $X.u$, define growth restricted roles. In general, the $Decompose$ technique does not remove statements as does COI, but instead constructs a set of RCPIs, each with a subset of statements from the original RCPI plus at most one new statement defining the temporary role $A'.r'$. This new statement is a replacement for an existing statement, such as statement 11 in Figure 4d is a replacement for statement 3 in Figure 4a.

One of the characteristics of this technique is that any sub-problem is no more expensive to analyze than the original RCPI since each sub-problem is effectively a subset of the original. We can express analysis effort primarily by the cardinality of the elements in SigRoles, as discussed in Section 2. It is beneficial to find counterexamples in simpler sub-problems in part due to the total analysis effort required, but also because a small sub-problem naturally narrows the scope of the counterexample, allowing a policy analyst to focus on a particular part of the policy for correction. In our example, we can identify a counterexample in Figure 4c where some principal E becomes a member of $Y.r$ but is not a member of $J.r$. A counterexample in this instance is easier to expose than in the instance of Figure 4d because only 16 new principals (4 significant roles, 2^4 principals) were sufficient to show satisfiability in this instance rather than 64 new principals (6 significant roles, 2^6 principals). Furthermore, the fact that a counterexample was found suggests that analysts may focus their policy correction attention on those roles and statements from this instance, as opposed to other roles and statements from the original policy.

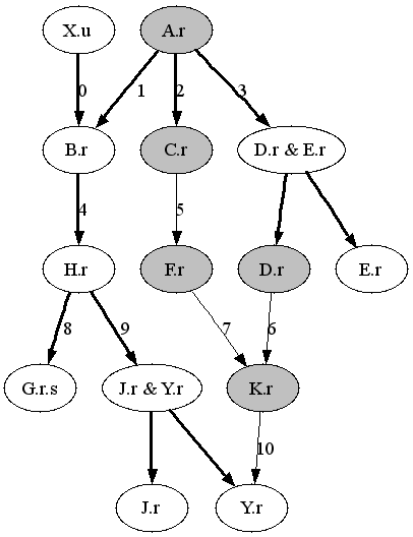
Now consider the example in Figure 5 where $\mathcal{P}$ is the initial policy, and the $Decompose$ technique constructs four sub-problem instances. This example differs from the previous in that $Decompose$ has been applied twice. It is not difficult to see that when we apply $Decompose$ once, one of the constructed sub-problem instances would be $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq G.r \rangle$ (not illustrated). We are able to apply $Decompose$ once more and produce two additional instances, as illustrated by Figures 5b and c. Thus $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq G.r \rangle$ is satisfied if and only if $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq H.r \rangle$ and $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq I.r \rangle$ are satisfied. Thus we demonstrate how this technique can be iteratively applied to produce potentially simpler RCPIs.

We now formally describe this reduction. $Decompose$ constructs the collection of new RCPIs.

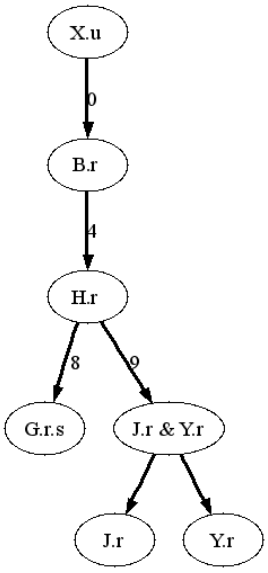
Index	Policy Statements		
0	$X.u \leftarrow B.r$	6	$D.r \leftarrow K.r$
1	$A.r \leftarrow B.r$	7	$F.r \leftarrow K.r$
2	$A.r \leftarrow C.r$	8	$H.r \leftarrow G.r.s$
3	$A.r \leftarrow D.r \cap E.r$	9	$H.r \leftarrow J.r \cap Y.r$
4	$B.r \leftarrow H.r$	10	$K.r \leftarrow Y.r$
5	$C.r \leftarrow F.r$	11	$A'.r' \leftarrow D.r \cap E.r$

$$\mathcal{G}_{\mathcal{R}} = \{A.r, C.r, D.r, F.r, K.r\},$$
$$\mathcal{S}_{\mathcal{R}} = \{A.r, B.r, H.r, X.u\}$$
$$\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$$

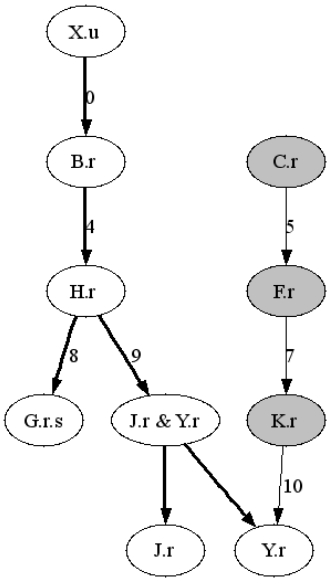
Policy State	Statement Indices
π	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
π_1	0, 4, 8, 9
π_2	0, 4, 5, 7, 8, 9, 10
π_3	0, 4, 6, 8, 9, 10, 11



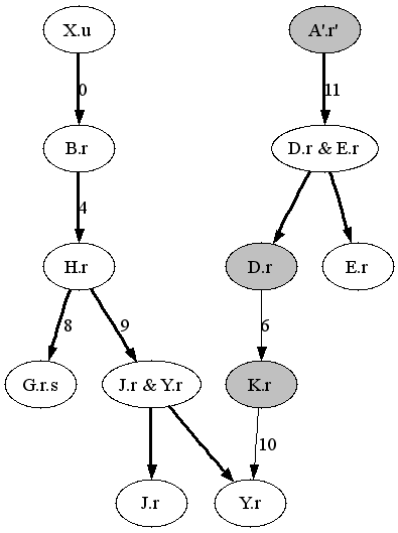
a. π



b. π_1

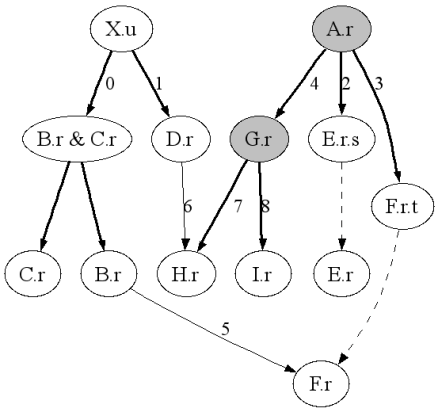


c. π_2

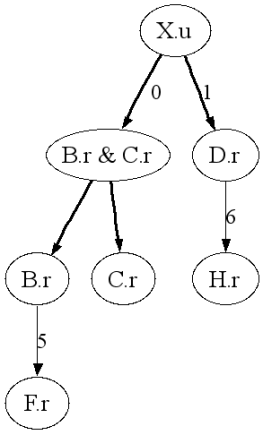


d. π_3

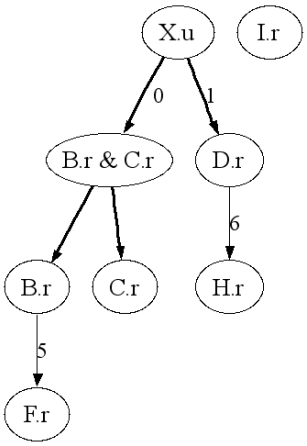
Figure 4: Decomposition example 1



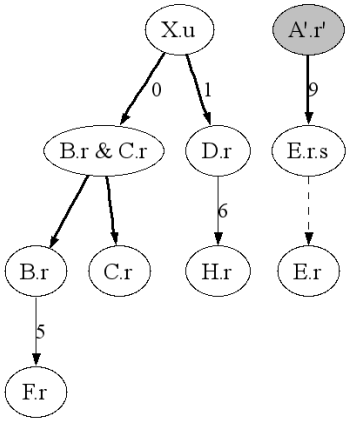
a. $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$



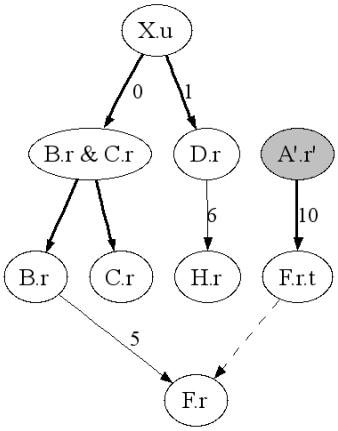
b. $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq H.r \rangle$



c. $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq I.r \rangle$



d. $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq A'.r' \rangle$



e. $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq A'.r' \rangle$

Figure 5: Decomposition example 2

Definition 12 *Given an RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, let A' be a new principal not in $\mathcal{P}$ and r' be a new role name not in $\mathcal{P}$. We define* $Decompose(\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle) = \{\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq \rho \rangle \mid A.r \leftarrow \rho \in \mathcal{P} \wedge \rho \in \mathsf{Roles}\} \cup \{\langle \mathcal{P} \cup \{A'.r' \leftarrow e\}, \langle \mathcal{G}_\mathcal{R} \cup \{A'.r'\}, \mathcal{S}_\mathcal{R} \cup \{A'.r'\} \rangle, X.u \sqsupseteq A'.r' \rangle \mid A.r \leftarrow e \in \mathcal{P} \wedge e \notin \mathsf{Roles}\}$.

Note that each of the new policies contains at most one occurrence of $A'.r'$, and that there is no occurrence of either A' or r' in any statement body. Furthermore, when e is a role, we do not make use of the new role $A'.r'$. We now formally describe the relationship between the solution to the original RCP instance and the solutions of the new problems.

Theorem 13 *Given an RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, if $A.r \in \mathcal{G}_\mathcal{R} \cap \mathcal{S}_\mathcal{R}$, then $\mathcal{P}$ satisfies $X.u \sqsupseteq A.r$ under $\mathcal{R}$ if and only if $\mathcal{P}'$ satisfies $X.u \sqsupseteq \rho$ under $\mathcal{R}'$ for each $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq \rho \rangle \in Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$.*

Observe that $Decompose$ may be applied iteratively to each new RCPI and potentially constructing even more new RCPIs. Assume the size of the policy $\mathcal{P}$ is given by n, the number of roles in $\mathcal{R}$ is k, and the set of roles in $\mathcal{R}$ are stored in a balanced tree data structure constructed in $O(k \log k)$ time with a look up time

of $O(\log\ k)$. A single iteration of this reduction costs $O(k\ \log\ k + n\ \log k)$ in general and $O(n\ \log\ n)$ when the set of roles in $\mathcal{R}$ is a subset of the roles of $\mathcal{P}$, which is typically the case. This single iteration takes a single subset query role $A.r$ and examines at most $O(n)$ dependant roles and test each of these roles for inclusion in the set $\mathcal{R}$. Thus the total cost of iteratively applying this reduction is $O(n^2\ \log\ n)$ since we examine at most $O(n)$ subset query roles.

In summary, $Decompose$ is a technique that may allow an analyst to de-construct a given RCPI into a set of potentially simpler RCPI sub-problems. The goal is to find a counterexample in a simpler sub-problem in the case that verifying the original problem is beyond ones computing resources. We demonstrate that this technique provides three benefits. First, while none of the sub-problems are guaranteed to be trivial, we can guarantee that each sub-problem is no more difficult to verify than the original problem. Second, in the case that each sub-problem can be shown to be satisfied, then we can be assured that the original problem is also satisfied. Consequently, if an analysis technique can show that any sub-problem is not satisfied, then we can be assured that the original problem cannot be satisfied. This is significant because even though an analysis technique may not be able to determine satisfiability a sub-problem due to a lack of resources, there may exist another sub-problem that can be shown as unsatisfied and within resource constraints. In such a case, we can conclude that the original problem is unsatisfiable. Finally, this technique can be applied iteratively to further de-construct sub-problems into potentially even simpler problems.

3.4 Other Possible Reductions

It seems likely that a further reduction could be defined and verified that would have the potential to reduce the number of principals in the policy part of the RCPI. Intuitively, a policy that has a lot of principals in it would be likely to have an equivalence relation over principals that are used in exactly the same way. We have not invested significant effort in determining exactly how that equivalence relation should be defined because, while theoretically interesting, in our view the result would be unlikely to be of great practical significance. The principals it seems likely could be eliminated from the policy by such a reduction are likely to be added through direct member statements to a handful of roles the membership of which would change frequently (*e.g.*, employee or student roles). Such roles are unlikely to be defined as growth- or shrink-restricted exactly because they must be modified frequently. As such, the direct member statements would be eliminated by the COI reduction.

We have omitted discussion of another reduction that would be simple to define and to verify. This reduction recognizes via purely syntactic means statements that have no effect on role membership. For instance if we have $A.r \longleftarrow B.r_1$ in the policy, there is no point in also including $A.r \longleftarrow B.r_1 \cap C.r_2$. These reductions are omitted here because they are straightforward and unilluminating.

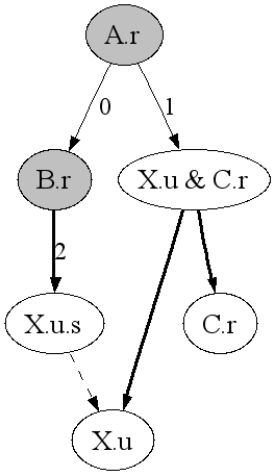
4 Model Construction

This section first presents corollaries, definitions, lemmas, and a theorem that show it is sufficient to explore a smaller policy state space than is justified by prior results or by those introduced above. These reductions differ from those in section 3 in that they do not transform one RCPI into another, but are more low-level. In particular, they identify individual statements in the reachable policy states that need not be considered while retaining a sound and complete evaluation of the original RCPI. The section then proceeds to construct a Finite State Machine (FSM) from an RCPI, which we call the *Analysis Finite State Machine* (AFSM), which can be used by the model checker to perform the analysis itself. Having presented an AFSM that can be used in the general case, we discuss a specialized version of the AFSM that can be applied to acyclic policies

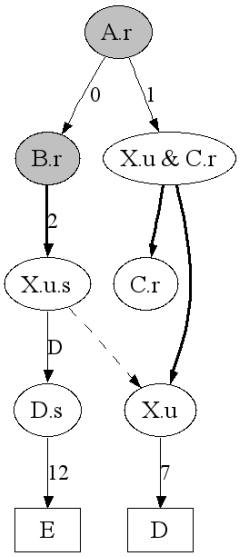
Index	Policy Statement of $\mathcal{P}$
0	$A.r \leftarrow B.r$
1	$A.r \leftarrow X.u \cap C.r$
2	$B.r \leftarrow X.u.s$

$\mathcal{G}_\mathcal{R} = \{A.r, B.r\}$
$\mathcal{S}_\mathcal{R} = \{B.r\}$

a. $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$



b. Visualization of π



c. Counterexample based on $\mathcal{N}(\pi)$

Figure 6: Running example

exploiting opportunities for significant performance improvement in this case.

4.1 Policy State-space Reductions

A naive implementation based directly on Theorem 6 would consider all policy states $\mathcal{P}'$ satisfying $\mathcal{P}' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$ and $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$. This section seeks to reduce the number and size of the policies that need to be considered while preserving the soundness and completeness of the security analysis. We begin with a corollary that simplifies slightly the statement of Theorem 6 by eliminating the reference to reachability.

Corollary 14 *Given an RCPI, $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, there exists a $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$ and there exists a principal E such that $E \in [\![A.r]\!]_{\mathcal{P}'}$ and $E \notin [\![X.u]\!]_{\mathcal{P}'}$ if and only if there exists a $\mathcal{P}''$ such that $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}'' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$ and there exists a principal E' such that $E' \in [\![A.r]\!]_{\mathcal{P}''}$ and $E' \notin [\![X.u]\!]_{\mathcal{P}''}$.*

Proof. For the "if" part, we need only show that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$. This follows from Definition 2 and the observation that $(\mathcal{N}(\pi)\restriction_{\mathcal{G}_\mathcal{R}}) = \emptyset$.

For the "only if" part, we show that the $\mathcal{P}''$ that is guaranteed to exist by Theorem 6 satisfies $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}'' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$. From the theorem, we have that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$ and $\mathcal{P}'' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$. By Definition 2 $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$ implies $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}''$, as required to complete the proof. ∎

Consider the example RCPI π given in Figure 6a. When we construct $\mathcal{N}(\pi)$, we calculate $\mathsf{SigRoles}(\pi) = \{X.u, C.r\}$, so $\mathsf{NewPrinc}(\pi)$ contains 2^2 new principals. Letting $\mathsf{NewPrinc}(\pi) = \{D, E, F, G\}$, the set of roles defined by $\mathcal{N}(\pi)$, given by $(\{A, B, C, D, E, F, G, X\} \times \{r, s, u\})$, has 24 elements. When we remove the growth restricted roles $\{A.r, B.r\}$, we are left with 22 growth unrestricted roles to build simple member statements. This construction yields $|\,\mathcal{N}(\pi)\,| = 176$ new simple member statements. This number, along with the original 3 statements of $\mathcal{P}$, produces an upper bound of 179 on the number of statements in the

policy states that must be explored, and because there is one statement defining a shrink restricted role, we have to examine at most 2^{178} reachable policy states.

Turning now to our goal of reducing the size of the state space that must be explored, we show that $\mathcal{N}(\pi)$ often contains a large number of simple member statements that need not be considered. In particular, the effect of many simple member statements constructed by using principals in $\mathsf{Principals}(\mathcal{P})$ can be simulated by simple member statements that instead use principals in $\mathsf{NewPrinc}(\mathcal{P})$. We illustrate this point by making three observations based on the example pictured in Figure 6.

First, a significant number of new simple member statements are introduced, defining eight sub-linked roles $A.s$, $B.s$, $C.s$, $D.s$, $E.s$, $F.s$, $G.s$, $X.s$. The construction produces 8^2 statements based on these roles. However, many of these roles function equivalently with respect to their capacity to generate a reachable policy state that violates the query. In this example, one growth unrestricted sub-linked role is sufficient for this purpose. While in general we have been unable to reduce the number of sub-linked roles down to one, and we are doubtful that doing so is safe in the general case, we do show that many statements defining such roles can safely be eliminated. Second, notice that the counterexample presented from the verification effort can be constructed with far fewer principals. The counterexample in Figure 6b uses only two principals. In particular, principals in $\mathsf{Principals}(\mathcal{P})$ that cannot form the linkage in a linked role can often be safely be excluded from simple member statements, as new principals can serve the same function in generating potential counterexamples. Third, the construction above produces several simple member statements such as $A.u \leftarrow B$ and $X.r \leftarrow C$ that define roles that the memberships of queried roles do not depend upon.

In the following two sections, we take two steps to reduce the number of policy states that must be considered, in each case eliminating the need to consider several statements in $\mathcal{P} \cup \mathcal{N}(\pi)$. In the first section, we construct $\mathcal{N}'(\pi) \subseteq \mathcal{N}(\pi)$ that as we prove, can be used in place of $\mathcal{N}(\pi)$ in defining the upper bound on policy states that must be considered. $\mathcal{N}'(\pi)$ limits the number of sub-linked roles that are considered, as well as the number of principals in $\mathsf{Principals}(\mathcal{P})$ that are used to construct simple member statements. In the second section, we eliminate from consideration policy statements that cannot affect the membership of queried roles. Finally, a theorem demonstrates the soundness of these reductions.

4.1.1 Constraining Simple Member Statement Principals

As illustrated above, it is often the case that we need not consider new statements defining sub-linked roles owned by certain principals in $\mathsf{Principals}(\mathcal{P})$. These statements are omitted from $\mathcal{N}'(\pi)$, which is constructed just below in Definition 15. Specifically, it is unnecessary to consider statements of the form $F.r_1 \longleftarrow e$ when $r_1 \in \mathsf{LinkedRoleNames}(\mathcal{P})$, in which $\mathsf{LinkedRoleNames}(\mathcal{P}) = \{r_1 \mid \exists\lambda.\exists B.\exists r.\lambda \leftarrow B.r.r_1 \in \mathcal{P}\}$, provided F is never introduced as a member of any role and $F.r_1$ has no statement defining or referencing it in $\mathcal{P}$. It is also unnecessary to consider statements of the form $\lambda_1 \longleftarrow F$ when F is never introduced as a member of any role and $F.r_1$ has no statement defining or referencing it in $\mathcal{P}$ for *any* $r_1 \in \mathsf{LinkedRoleNames}(\mathcal{P})$. Intuitively, these are cases in which $F.r_1$ cannot participate in contributing to the member of λ via the statement $\lambda \longleftarrow B.r.r_1$. Before defining $\mathcal{N}'(\pi)$, we recall that for $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, $\mathsf{NewPrinc}(\pi)$ is a set of new principals, disjoint from $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{Principals}(\mathcal{R})$, and whose cardinality is of size $2^{|SigRoles(\pi)|}$ in which $\mathsf{SigRoles}(\pi)$ is the set $\{X.u\} \cup \{B_1.r_1 \mid B.r \longleftarrow B_1.r_1.r_2 \in \mathcal{P}\} \cup \{B_1.r_1, B_2.r_2 \mid B.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P}\}$ and $\mathsf{Principals}(\mathcal{R})$ is the set of principals in the restriction rule.

Definition 15 *Given any RCPI, $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, define $\mathcal{N}'(\pi)$ as follows.*

$$\begin{aligned}
\mathcal{N}'(\pi) =& \{A.r \longleftarrow D \mid A.r \notin \mathcal{G}_{\mathcal{R}} \wedge r \in \mathsf{Names}(\mathcal{P}) \wedge \\
& A, D \in \mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi) \wedge \\
& A.r \notin \mathsf{NoLinkedDefs}(\mathcal{P}) \wedge D \notin \mathsf{NoLinkedPrinc}(\mathcal{P})\} \\
\mathsf{NoLinkedDefs}(\mathcal{P}) =& \{F.r_1 \mid F \in \mathsf{Principals}(\mathcal{P}) \wedge r_1 \in \mathsf{LinkedRoleNames}(\mathcal{P}) \wedge \\
& F.r_1 \notin \mathsf{ConcreteRoles}(\mathcal{P}) \wedge \forall \lambda \in \mathsf{ConcreteRoles}(\mathcal{P}).\lambda \longleftarrow F \notin \mathcal{P}\} \\
\mathsf{LinkedRoleNames}(\mathcal{P}) =& \{r_1 \mid \exists \lambda. \exists B. \exists r. \lambda \leftarrow B.r.r_1 \in \mathcal{P}\} \\
\mathsf{NoLinkedPrinc}(\mathcal{P}) =& \{F \mid F \in \mathsf{Principals}(\mathcal{P}) \wedge \\
& \forall r_1 \in \mathsf{LinkedRoleNames}(\mathcal{P}).F.r_1 \notin \mathsf{ConcreteRoles}(\mathcal{P}) \wedge \\
& \forall \lambda \in \mathsf{ConcreteRoles}(\mathcal{P}).\lambda \longleftarrow F \notin \mathcal{P}\} \\
\mathsf{ConcreteRoles}(\mathcal{P}) =& \{B.r \mid \exists e_1.B.r \leftarrow e_1 \in \mathcal{P} \vee \\
& \exists \lambda \leftarrow e_2 \in \mathcal{P}.e_2 \in \{B.r, B.r.r_1, B.r \cap C.r_1, C.r_1 \cap B.r\}\}
\end{aligned}$$

Intuitively, $\mathsf{ConcreteRoles}(\mathcal{P})$ is the set of roles appearing in $\mathcal{P}$; $\mathsf{LinkedRoleNames}(\mathcal{P})$ is the set of role names that appear as the second role name in a linked role $B.r.r_1$ occurring in $\mathcal{P}$; $\mathsf{NoLinkedDefs}(\mathcal{P})$ is the set of roles $F.r_1$ not appearing in $\mathcal{P}$ such that F and r_1 do appear in $\mathcal{P}$, $r_1 \in \mathsf{LinkedRoleNames}(\mathcal{P})$, and F is not a member of any role in $\mathcal{P}$; $\mathsf{NoLinkedPrinc}(\mathcal{P})$ is the set of principals in $\mathcal{P}$ that cannot contribute to the membership of a linked role $B.r.r_1$ in $\mathcal{P}$ as a result of being a member of $B.r$. Finally, $\mathcal{N}'$ is obtained from $\mathcal{N}$ by removing statements $A.r \longleftarrow D$ such that $A.r \notin \mathsf{NoLinkedDefs}(\mathcal{P}) \wedge D \notin \mathsf{NoLinkedPrinc}(\mathcal{P})$. The intuition made precise by the following lemma is that the roles played by such statements in constructing a counterexample can be played by statements constructed by using principals not appearing in $\mathcal{P}$.

Consider the running example RCPI π given in Figure 6a. We construct $\mathcal{N}'(\pi)$ by assuming that $\mathsf{NewPrinc}(\pi) = \{D, E, F, G\}$. By applying the definition, we obtain $\mathsf{ConcreteRoles}(\mathcal{P}) = \{A.r, B.r, C.r, X.u\}$, $\mathsf{NoLinkedDefs}(\mathcal{P}) = \{A.s, B.s, C.s, X.s\}$, and $\mathsf{NoLinkedPrinc}(\mathcal{P}) = \{A, B, C, X\}$, which is equivalent to $\mathsf{Principals}(\mathcal{P})$. The set of roles to construct simple member statements defined by $\mathcal{N}'(\pi)$ contains 18 elements by removing the 4 elements in $\mathsf{NoLinkedDefs}(\mathcal{P})$ and 2 growth restricted roles $\{A.r, B.r\}$ from the set of roles given by $\{A, B, C, X, D, E, F, G\} \times \{r, s, u\}$. The set of principals to construct simple member statements contains only the 4 elements in $\mathsf{NewPrinc}(\pi)$. Thus, the construction yields 18 $\times$ 4 new statements, in addition to the original 3 statements of P in π.

Lemma 16 (Equivalence of using $\mathcal{N}$ and $\mathcal{N}'$) *Let $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ be any RCPI. There exists $\mathcal{P}'$ and $E \in \mathsf{Principals}(\mathcal{P}')$ such that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$, $E \in [\![A.r]\!]_{\mathcal{P}'}$, and $E \notin [\![X.u]\!]_{\mathcal{P}'}$ if and only if there exists $\mathcal{P}''$ and $E \in \mathsf{Principals}(\mathcal{P}'')$ such that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$, $E \in [\![A.r]\!]_{\mathcal{P}''}$, and $E \notin [\![X.u]\!]_{\mathcal{P}''}$.*

Note that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$ implies $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}''$.

4.1.2 Reducing the Number of Roles that Need to be Considered

Lemma 16 tells us that to determine whether an RCPI π is satisfied it is sufficient to look for counterexamples $\mathcal{P}'$ satisfying $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$. It is often possible to substantially further reduce the number

of policy states that must be considered by taking advantage of the fact that many policy statements cannot affect the memberships of the queried roles. This section shows how to take advantage of this opportunity.

To do this, we make use of a construction very similar to that of DefRoles. In the policy states $\mathcal{P}''$ under consideration in Corollary 14, the set of roles on which queried-role membership can depend includes sub-linked roles that are constructed from new principals not appearing in $\mathcal{P}$. The set of roles we require is given by $\mathsf{DefRoles}'$, which we now define.

Definition 17 *Let $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ be any RCPI. We denote by $\mathsf{DefRoles}'(\pi)$ the least set of roles $\mathcal{O}$ satisfying the following conditions:*

- $X.u, A.r \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1 \in \mathcal{P}) \Rightarrow B.r_1 \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1.r_2 \in \mathcal{P} \wedge D \in \mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)) \Rightarrow (B.r_1 \in \mathcal{O} \wedge D.r_2 \in \mathcal{O}))$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \leftarrow B.r_1 \ \cap \ C.r_2 \in \mathcal{P}) \Rightarrow (B.r_1 \in \mathcal{O} \wedge C.r_2 \in \mathcal{O})$

As shown below, it is sufficient to consider only policy states consisting of statements that define roles in $\mathsf{DefRoles}'(\pi)$. In fact, it may not be necessary to consider the initial policy state itself if it contains statements defining roles on which the queried roles do not depend.

Consider again the RCPI depicted in Figure 6. In this case, $\mathsf{DefRoles}'(\pi) = \{A.r, B.r, C.r, X.u, A.s, B.s, C.s, D.s, E.s, F.s, G.s, X.s\}$. This reduces from 24 to 12 the number of roles whose definitions need to be considered. These 12 roles reduce by half the 24 required roles by the original construction $\mathcal{N}(\pi)$.

Lemma 18 (Projecting onto $\mathsf{DefRoles}'$) *Let $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ be any RCPI. There exists $\mathcal{P}'$ and $E \in \mathsf{Principals}(\mathcal{P}')$ such that $E \in [\![A.r]\!]_{\mathcal{P}'}$, $E \notin [\![X.u]\!]_{\mathcal{P}'}$, and $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}''' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$, if and only if there exists $\mathcal{P}''$ and $E \in \mathsf{Principals}(\mathcal{P}'')$ such that $E \in [\![A.r]\!]_{\mathcal{P}''}$, $E \notin [\![X.u]\!]_{\mathcal{P}''}$, and $E \in [\![A.r]\!]_{\mathcal{P}''}$, $E \notin [\![X.u]\!]_{\mathcal{P}''}$, and $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R} \cap \mathsf{DefRoles}'(\pi)} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$.*

Proof. It is easily shown by induction on the steps of evaluation of $\mathcal{P}'$ and $\mathcal{P}'\restriction_{\mathsf{DefRoles}'(\pi)}$ that the membership of roles in $\mathsf{DefRoles}'(\pi)$ are identical. ∎

Theorem 19 *Given an RCPI, $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, there exists a $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$ and there exists a principal E such that $E \in [\![A.r]\!]_{\mathcal{P}'}$ and $E \notin [\![X.u]\!]_{\mathcal{P}'}$ if and only if there exists $\mathcal{P}''$ and $E \in \mathsf{Principals}(\mathcal{P}'')$ such that $E \in [\![A.r]\!]_{\mathcal{P}''}$, $E \notin [\![X.u]\!]_{\mathcal{P}''}$, and and $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R} \cap \mathsf{DefRoles}'(\pi)} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$.*

Proof. Follows by applying Corollary 14, Lemma 16, and Lemma 18. ∎

Note that because of the projection onto $\mathsf{DefRoles}'(\pi)$, this theorem enables us consider policy states for which it is not necessarily the case that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$. Thus, $\mathcal{P}''$ is not generally a counterexample for the original RCPI, π. Theorem 19 tells us only that a counterexample for π exists if and only if a $\mathcal{P}''$ exists satisfying the requirements in the theorem. This is helpful because the size of the policy states that must be examined in finding $\mathcal{P}''$ is generally smaller than if we omit the restriction to $\mathsf{DefRoles}'(\pi)$. This makes model checking more likely to be successful. Reachable states that differ only in the definition of roles outside $\mathsf{DefRoles}'(\pi)$ do not have to be considered separately from one another. However, we wish to present to the user not only information as to whether a counterexample exists. When one does exist, we would like to exhibit it so that the user can identify the source of the problem. The following proposition enables us to do so.

Proposition 20 *Given a $\mathcal{P}''$ satisfying the requirements stated in Theorem 19, we can construct $\mathcal{P}' = \mathcal{P}'' \cup \mathcal{P}\restriction_{\mathcal{S}_\mathcal{R} - \mathsf{DefRoles}'(\pi)}$, which satisfies $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}'$, $[\![A.r]\!]_{\mathcal{P}'} = [\![A.r]\!]_{\mathcal{P}''}$, and $[\![X.u]\!]_{\mathcal{P}'} = [\![X.u]\!]_{\mathcal{P}''}$, from which it follows that $\mathcal{P}'$ satisfies the requirements stated in Theorem 19.*

Proof. The proposition follows by induction on the construction of $T_\mathcal{P} \uparrow^\omega [\![B.r_1]\!]$ and the definition of $\mathsf{DefRoles}'(\pi)$. ∎

Thus we can return the $\mathcal{P}'$ identified in Proposition 20 to the user as a counterexample to π when one exists. The following definition formalizes the policy states that need to be model checked to determine whether the query is satisfied by every state that is reachable from π. We say that this set is *to be evaluated (TBE).*

Definition 21 ($\mathsf{TBE}(\pi)$ and $\mathsf{MaxTBE}(\pi)$) *Given an RCPI π, we define the set of policy states* to be evaluated for π *by* $\mathsf{TBE}(\pi) = \{\mathcal{P}' \mid \mathcal{P}\restriction_{\mathcal{S}_\mathcal{R} \cap \mathsf{DefRoles}'(\pi)} \subseteq \mathcal{P}' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}\}$. *We call the largest policy state in* $\mathsf{TBE}(\pi)$, $\mathsf{MaxTBE}(\pi) = (\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$.

Let us once again reconsider the example depicted in Figure 6. We obtain 75 statements in $\mathcal{P} \cup \mathcal{N}'(\pi)$. Projecting onto $\mathsf{DefRoles}'(\pi)$, we obtain 27 statements in $(\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$. Because $\mathcal{N}'(\pi)$ contains no statements defining roles in $\{A.s, B.s, C.s, X.s\}$, $\mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$ defines only 8 roles: $\{A.r, B.r, C.r, X.u, D.s, E.s, F.s, G.s\}$. Of these, $A.r$ and $B.r$ are growth restricted. Thus, only 6 roles and 4 principals will be used to construct new statements, so the total number of statements in $\mathcal{N}'(\pi)\restriction_{\mathsf{DefRoles}'(\pi)}$ would be 24, making the number of policy statements in $(\mathcal{P} \cup \mathcal{N}'(\pi))\restriction_{\mathsf{DefRoles}'(\pi)}$ 27. This is a significant reduction from the 179 statements we would have been required to examine in the original construction.

4.2 Analysis Finite State Machine

Model checkers take as input a finite state machine (FSM) and a property specification expressed as a temporal-logic formula. They then determine whether the FSM satisfies the formula. We use the following standard definition of an FSM.

Definition 22 (FSM) *A* finite state machine *is given by the 3-tuple, $\langle S, S_0, \delta\rangle$, in which S is a finite, non-empty set of states, $S_0 \subseteq S$ is a set of initial states, and $\delta \subseteq S \times S$ is a transition relation.*

(Some standard definitions also include a labeling function that map states to structured objects, such as sets of propositions, interpretations, or variable bindings. We take the alternative of making states themselves be structured objects, as we discuss below.)

In this section we present two FSMs that can be used solve RCPIs. The first of these can be applied to any RCPI, π, and is denote by $\mathsf{AFSM}(\pi)$. This FSM uses state transitions to perform the fixpoint calculation that compute role memberships. (NuSMV automatically analyzes dependencies among role definitions and essentially replaces each use of a role in a statement by an expression derived from the definition of that role in the policy state, together with the role memberships that had been obtained in the prior step of the fixpoint calculation.) The second FSM we present can be applied safely only to RCPIs $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r\rangle$ in which $\mathcal{P}$ is non-recursive. We denote this latter FSM by $\mathsf{AFSMOPT}(\pi)$. In $\mathsf{AFSMOPT}(\pi)$, we simply generate each policy state in $\mathsf{TBE}(\pi)$ and rely on NuSMV to compute role membership efficiently in a single step. Because $\mathcal{P}$ is non-recursive, this first role membership assignment is guaranteed to be the fixpoint. Consequently, $\mathsf{AFSMOPT}(\pi)$ need not take a transition simulating an iteration of the fixpoint calculation, nor determine that the fixpoint has been reached. Below in Section 5, we present how these abstract structures are expressed in the NuSMV FSM specification language.

$$
\begin{array}{rcl}
\mathrm{S} & = & \mathsf{TBE}(\pi) \times \beta \times \{\mathit{fix}, \mathit{eval}\} \\
\beta & = & \mathsf{ConcreteRoles}(\mathsf{MaxTBE}(\pi)) \rightarrow \wp(\mathsf{Principals}(\mathsf{MaxTBE}(\pi))), \\
 & & (\beta, \text{the set of role membership functions, is the set of functions from roles to} \\
 & & \text{sets of principals.}) \\
S_0 & = & \mathsf{TBE}(\pi) \times \{\eta_0\} \times \{\mathit{eval}\} \\
\eta_0(\lambda) & = & \emptyset, \text{ for each } \lambda \in \mathsf{ConcreteRoles}(\mathsf{MaxTBE}(\pi)) \\
 & & (\eta_0, \text{the initial role membership function, maps each role to the empty set of principals}) \\
\delta & = & \{\langle\langle \mathcal{P}_1, \eta, \mathit{eval}\rangle, \langle \mathcal{P}_1, \eta', \mathit{eval}\rangle\rangle \mid \eta' = T_{\mathcal{P}_1}(\eta) \wedge \eta' \neq \eta\} \cup \\
 & & \{\langle\langle \mathcal{P}_1, \eta, \mathit{eval}\rangle, \langle \mathcal{P}_1, \eta, \mathit{fix}\rangle\rangle \mid \eta = T_{\mathcal{P}_1}(\eta)\} \\
 & & \text{(When applying } T_{\mathcal{P}_1}(\eta) \text{ yeilds a new role membership function, we keep evaluating;} \\
 & & \text{otherwise a fixpoint has been reached.)}
\end{array}
$$

Figure 7: Construction of $\mathsf{AFSM}(\pi)$

4.3 General Case

The specification of $\mathsf{AFSM}(\pi)$ is given in Figure 7 in terms of the abstract structures of the form given in Definition 22. It evaluates the role membership of each policy state in $\mathsf{TBE}(\pi)$ for arbitrary RCPIs π. Each state of $\mathsf{AFSM}(\pi)$ include not only a given policy state in $\mathsf{TBE}(\pi)$, but also a role membership function $\eta \in \beta$, which maps each role to a set of principals in that role. It also include a flag that takes on either the value *fix* or eval, which we discuss next. Because the RT language allows cyclic dependencies among roles, we found it necessary to use a sequence of state transitions to calculate the fixpoint that determines the membership of each role. This is accomplished by differentiating between what we call *fixpoint mode* (*fix*) and *evaluation mode* (eval). Mode eval signifies that the calculation of role memberships has not yet stabilized to a fixpoint, so the query should not be evaluated yet; from such a state, the fixpoint evaluation proceeds with the evaluation process. Mode *fix* signifies that the fixpoint has been reached (none of the role memberships change between successive states), and that the role membership function now accurately reflects the membership of each role in the given policy state. The membership of each role under the given policy is now known and the query can now be evaluated. The fact that the query should be evaluated only after a fixpoint has been reached is encoded in the CTL formula that is evaluated with respect to runs of $\mathsf{AFSM}(\pi)$. (See Section 5.)

Figure 8 illustrates two concepts. First, observe that the all the non-determinism occurs in the selection of an initial state, which corresponds to the selection of the policy in $\mathsf{TBE}(\pi)$ that is evaluated. These are represented by the policy states pictured as $\mathcal{P}_0 \ldots \mathcal{P}_{n-1}$ in which $n = |\mathsf{TBE}(\pi)|$. States in eval model are represented by a gray dots. Transitions out of these states represent steps in the evaluation of the fixpoint. Second, the states in *fix* mode (represented by black dots) have no out transitions. These are the states in which the query is actually evaluated.

4.4 Non-Recursive Case

Let us now definite $\mathsf{AFSMOPT}(\pi)$, which is can be safely applied only when π is non-recursive, but achieves significant savings in this case. This construction is given in Figure 9. In this case, the FSM state contains only the element of $\mathsf{TBE}(\pi)$ to be evaluated. The role membership function is calculated by the CTL formula (see Section 5).

The key difference between the general and the non-recursive versions is that the latter can define the membership of any role λ as a formula where each dependent role $B.r$ in the formula is replaced with the

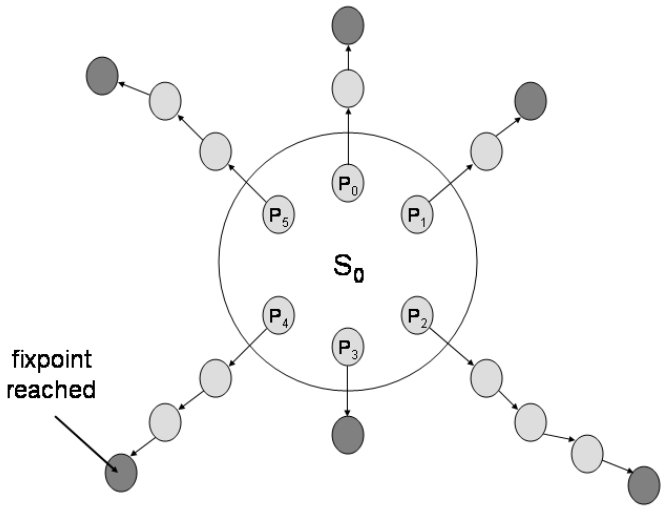


Figure 8: Relation of policy states to evaluation states for an RCPI in which $|\mathsf{TBE}(\pi)| = 6$

$$\begin{array}{rcl} S & = & \mathsf{TBE}(\pi) \\ S_0 & = & S \\ \delta & = & \emptyset \end{array}$$

Figure 9: Construction of $\mathsf{AFSM}(\pi)$ Optimized for the case in which $\mathcal{P}$ is Non-recursive where $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$

definition of $B.r$. Since there are no cyclic dependencies, the definition of λ can be resolved to an expression with a known and finite number of terms, where each of the terms is a Boolean indicating the presence or absence of a policy statement in the current policy state. Thus for any given role λ and principal C, C is in λ if certain policy statements are present.

5 Model Checking RT Policy

This section describes the translation process that takes as input an RCPI π and $\mathsf{MaxTBE}(\pi)$, and builds an SMV model and a property specification (written in temporal logic) that expresses the policy analysis query. The SMV model and specification are passed to the SMV model checker and the results of the model checking are returned to the user in the form of an affirmation or a counterexample. We describe two translation processes to handle both the general case and the optimized non-recursive case.

Model checking is an automated verification technique to verify finite systems, usually modeled as FSMs. Model checking exhaustively explores the state space based on the transition relation of a finite system to determine if a given property, usually expressed in temporal logic, hold. Temporal logic is a specification language for expressing properties related to sequences of states in terms of temporal operators and logic connectives. In the case that a property fails to hold, a model checker can produce a counterexample consisting of a trace that shows how the failure can arise, which can be used to correct the model or the

property specification. We choose to use Cadence SMV because it is a general purpose model checker and it is a BDD-based (symbolic), highly-optimized tool that can handle a relatively large state space.

5.1 General Translation

Once we have constructed the $\mathsf{MaxTBE}(\pi)$ for a given RCPI π or an RCPI π created by reduction techniques (*e.g.*, decomposition), we translate the (π, $\mathsf{MaxTBE}(\pi)$) pair into a model and a property specification in SMV's input language. Translation consists of four steps, each corresponding to the construction of one of the core components input to the SMV model checker: data structure declaration, initialization, transition relation, and property specification.

We briefly describe SMV by focusing on the features that we use. SMV models are FSMs. States are given by assignments of values to state variables, which must be of finite type. Variables are declared at the beginning of the model definition using the syntax $\langle variable \rangle : \langle type \rangle$ where $variable$ is an identifier that refers to the name of a data structure and $type$ refers to one of finite types such as boolean, an enumerated type, or a range of integers.

The initial states are defined by using $init$ statements of the form $init(x) := exp$, which defines the value or set of values x can take on initially. Transition relation is represented by using a collection of $next$ statements of the form $next(x) := exp$, which defines the value or set of values that x can assume in the following state (by taking the transition in the current state). Transitions can be non-deterministic; SMV explores all possible combinations of such choices. In a $next$ statement, exp can refer to the values of variables in the current state (*e.g.*, x) or in the next state (*e.g.*, $next(x)$). However, there must not be cyclic dependencies among values in the next state. In particular, iteration cannot be used to define the transition relation. (This becomes relevant in our context in the evaluation of role memberships when policies contain cyclic dependencies among roles.) On the other hand, conditional statements *can* be used to specify transitions, in the form $if(\langle condition \rangle)\,\{\langle body1 \rangle\}$ or $if(\langle condition \rangle)\,\{\langle body1 \rangle\}\,else\,\{\langle body2 \rangle\}$, where $\langle condition \rangle$ is a Boolean expression, and $\langle body1 \rangle$ and $\langle body2 \rangle$ are sets of next statements. The next state is computed from the current state by executing each $next$ statement (in an order that respects the dependencies among them). SMV supports derived variables, which are essentially macros that expand to expressions over state variables. Derived variables are defined by using assignment statements of the form $x := exp$ and are not explicitly represented in any state. Instead, variable x is replaced by exp, which refers to state variables.

The property specification defines a Computation Tree Logic (CTL) formula to be checked against the SMV model. (CTL is a branching-time temporal logic). In our work we use the universal path quantifier and temporal operator *henceforth*, denoted as $\mathbf{AG}(\varphi)$, to express that the property φ holds in every state along all paths.

Comments are marked as those lines that begin with the double dash $(--)$.

5.1.1 Data Structures

Data structures in SMV hold state information of an FSM, which is utilized to compute the next state. Besides the simple/generic data types, we can also specify an array using the syntax $\langle variable \rangle : array\ \langle x \rangle..\langle y \rangle\ of\ \langle type \rangle$ where x and y specify the range of the array. A specific index can be assigned or referenced using the square brackets operator such as $\langle variable \rangle[z]$, where $variable$ is the name of an array and z is a non-negative integer. In our translation, we only utilize Boolean and Boolean array (bit vector) data structures.

```
-- ***Policy Model***
-- <0> A.r <-- B.r
-- <1> A.r <-- X.u & C.r
-- <2> B.r <-- X.u.s
-- <3> C.r <-- D
-- <4> C.r <-- E
-- <5> C.r <-- F
-- <6> C.r <-- G
...
-- <25> X.u <-- F
-- <26> X.u <-- G

-- ***Policy Principals***
-- <0> D
-- <1> E
-- <2> F
-- <3> G
```

a. Comments representing MaxTBE(π)

```
-- bit for each policy statement
s :  array 0..26 of boolean;

-- bit for each principal per role
Ar :  array 0..3 of boolean;
Br :  array 0..3 of boolean;
Cr :  array 0..3 of boolean;
Ds :  array 0..3 of boolean;
Es :  array 0..3 of boolean;
Fs :  array 0..3 of boolean;
Gs :  array 0..3 of boolean;
Xu :  array 0..3 of boolean;

-- boolean fixpoint
fixpoint :  boolean;
```

b. NuSMV data structures

Figure 10: Data structure declaration

Each AFSM declares three types of variables. The first represents the statements in MaxTBE(π), each of which is represented by a unique index of a bit vector named s. This type of declaration would have the form $s : array\ 0..m\ of\ boolean$ where $m = |\text{MaxTBE}(\pi)| - 1$. Each policy state in TBE(π) (*i.e.*, the set of states to be evaluated) can be represented by an assignment of bit vector s: the value of a vector element being $true$ indicates that the corresponding statement is present in that state.

The second type of state variable represents role membership. It consists of one bit vector for each role λ, in which each principal is represented in each vector by a fixed index. Thus for each role λ in the ConcreteRoles (MaxTBE(π)), this declaration has the form $\lambda : array\ 0..n\ of\ boolean$ where $n = |\text{Principals}(\text{MaxTBE}(\pi))| - 1$.

The AFSM uses a sequence of state transitions to calculate role memberships as a means to recursively evaluate an policy exhibiting cyclic dependencies. The third type of state variable is a Boolean flag indicating whether the role membership calculation has reached a fixpoint. This requires a declaration of the form $fixpoint : boolean$, where a value of $true$ indicates that the fixpoint has been reached and $false$ otherwise.

Suppose that we were to translate the (π, MaxTBE(π)) from the running example in Figure 6. Figure 10a shows its 27 statements, 8 roles, and 4 policy principals, which are shown as SMV comments. Figure 10b shows the result of translating it into SMV data structures.

5.1.2 Initial State

s is a bit vector that indicates which RT statements are present in the current policy state. The manner in which we initialize $s[i]$ depends on the statement represented by index i. The statements that define shrink restricted roles cannot be removed from any policy state. Thus, $s[i]$ is initialized by the statement $s[i] := 1$ if statement i defines a role in $\mathcal{S}_\mathcal{R}$, which means that this statement is included in any other policy states. (The value 1 represents $true$ in SMV, indicating that statement i is in the policy state.) This makes $s[i]$ a derived variable (*i.e.*, a constant), which does not contribute to the size of the state space. On the other hand, when statement i does not satisfy these constraints, it is present in some reachable policy states and not in

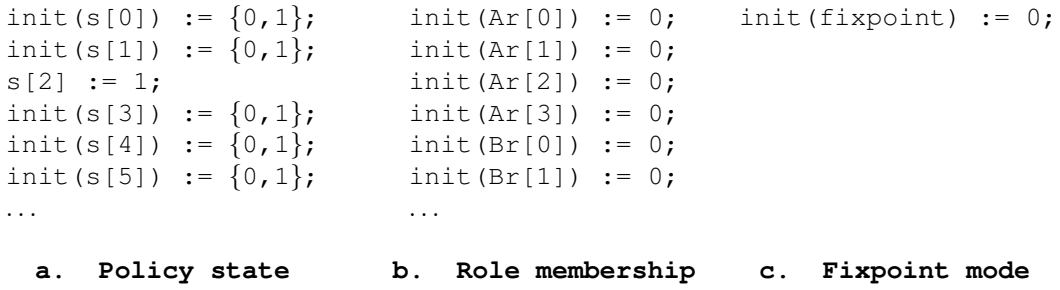


Figure 11: Initialization

others. In such a case, we initialize statement i of the form $init(s[i]) := \{0, 1\}$, which nondeterministically sets the value of statement i to $true$ or $false$. In this way, the set of initial states of the SMV model represent all possible policy states in $\mathsf{TBE}(\pi)$. (Recall that each policy state is a collection of policy statements.)

The bit vectors representing role membership are initialized to represent empty sets of principals. For each role λ and each principal indexed by j, we assign $init(\lambda[j]) := 0$. Finally, the $fixpoint$ flag is initialized to $false$ by assigning $init(fixpoint) := 0$.

Figure 11 illustrates how the initialization component of our SMV model for the running example would be expressed. Observe that we initialize the model to the set of reachable policy states and certain statements are included as a constant in every policy state. For example, $s[2]$ corresponds to statement $B.r \leftarrow X.u.s$ in the model's comment header, and is included in every reachable policy state because it was in $\mathcal{P}$ and $B.r \in \mathcal{S}_{\mathcal{R}}$. Furthermore, since we begin in a non-fixpoint state, we initialize the roles to the empty set of principals and assign $false$ to $fixpoint$.

5.1.3 Transition Relation

We now discuss how to implement the fixpoint calculation of role membership in SMV's input language. We represent the evaluation of role membership steps, each of which is implemented by a set of transitions updating the membership of all the roles (and the policy state remain fixed). When none of the transitions change the role membership in a state, the fixpoint has been reached, and the flag, $fixpoint$, is updated to $true$ by a transition. At this point, the FSM run is complete, the FSM stays in this state forever, which in implemented by the next statement of the form $next(s[i]) := s[i]$, and the query is checked. The CTL specification that checks query satisfaction ignores states in which the fixpoint has not yet been reached.

The detailed implementation of role membership calculation is described as below. Role membership in the next state is calculated from role membership in the current state based on the policy statements that are present in the current policy state. For each $\lambda \in \mathsf{ConcreteRoles}(\mathsf{MaxTBE}(\pi))$, and each principal (given by some $j \in [0..|\mathsf{Principals}(\mathsf{MaxTBE}(\pi))| - 1]$), an assignment of the form $next(\lambda[j]) := exp$ is introduced. The form of expression exp depends on the form of the policy statements that define λ. It is a disjunction of expressions containing one disjunct for each $\lambda \leftarrow e \in \mathsf{MaxTBE}(\pi)$. The form of this disjunct is given in figure 12 based on the form of e, in which i is the index used to represent this statement and "|" represents disjunction. The case of linked roles bears some discussion: principal j is added to λ if there is any principal C (indexed by k in the construction) such that in the previous state C is in $B.r$ and principal j is in $C.s$.

To maintain the $fixpoint$ flag correctly, the SMV input includes the statement $next(fixpoint) := exp$ where exp is the conjunction of expressions of the form $next(\lambda[j]) = \lambda[j]$ over all roles λ and all principals j. If the next state of each role is the same as its current state, the fixpoint of role membership calculation

Statement Type	Form	Translated Expression
Simple Member	$\lambda \leftarrow B$	$s[i]$
Simple Inclusion	$\lambda \leftarrow B.r$	$s[i]$ & $Br[j]$
Linked Inclusion	$\lambda \leftarrow B.r.s$	$s[i]$ & $(e_0 \mid \ldots \mid e_k \mid \ldots \mid e_m)$ $m = \lvert\mathsf{Principals}(\mathsf{MaxTBE}(\pi))\rvert - 1$, for each $k \in [0..m]$, $e_k = (Br[k]$ & $Cs[j])$, and C is the principal indexed by k
Intersection Inclusion	$\lambda \leftarrow B.r \cap C.s$	$s[i]$ & $Br[j]$ & $Cs[j]$

Figure 12: Construction of expressions for role membership in the next state

has been reached in the current state.

Figure 13 illustrates how we would construct the SMV model for our running example. We constrain the next policy state to the current state when flag $fixpoint$ is $false$ (evaluation mode). Note that policy statement variable $s[2]$ is not assigned any value since it is defined as a constant. Furthermore, role $A.r$ is defined by the statements represented by $s[0]$ and $s[1]$, and these definitions are combined using disjunction. In addition, role $B.r$ is defined by a linked inclusion statement and we express this in SMV by considering all of the sub-linked roles that could be formed by the base-linked role $X.u$. For instance, if $Xu[0]$ is $true$ in some state $\mathcal{P}'$, this implies that $D \in [\![X.u]\!]_{\mathcal{P}'}$ and the membership of $D.s$ may influence the membership of $A.r$. Finally, flag $fixpoint$ assumes the value $true$ only when all of the roles in the model have reached a fixpoint.

5.1.4 Property Specification

The role containment query, $X.u \sqsupseteq A.r$, is represented as a CTL formula, `AG (fixpoint -> ((Xu | Ar) = Xu))`, which asserts that in all reachable states, when $fixpoint$ is $true$ every member of A.r is also a member of X.u. The expression `Xu | Ar` performs bitwise-or and implements union, defines that the set of principals in `Ar` must be a subset of those in `Xu`.

5.2 Non-Recursive Case Translation

An alternative and optimized translation is developed for policies without cyclic dependencies. This translation results in savings that derive from two major factors. First, it eliminates the use of intermediate states and transitions associated with calculating role membership. Second, we can use derived variables, rather than state variables, to express role membership. Adding the derived variables will not increase the state space.

The data structures of the SMV model remain the same as those used in the general translation, with exception that there is no *fixpoint* variable and all role membership variables are derived variables (*i.e.*, macros). However, the initialization process is different. We still initialize the policy state as before, but the role membership variables, which are now derived variables, are not initialized. Instead we define how they are represented by using policy statement variables. When we write the specification in SMV as `AG ((Xu | Ar) = Xu)`, the derived variables (*e.g.*, `Xu`) in the specification are automatically replaced by their definitions, which refer to policy statement variables. In this way, the specification will be expressed solely in terms of policy statement variables.

```
if( fixpoint )
{
    -- stay in this state forever
    next(s[0]) := s[0];
    next(s[1]) := s[1];
    next(s[3]) := s[3];
    ...

    next(Ar[0]) := Ar[0];
    next(Ar[1]) := Ar[1];
    next(Ar[2]) := Ar[2];
    ...

    next(fixpoint) := fixpoint;
}
else
{
    -- next policy state
    next(s[0]) := s[0];
    next(s[1]) := s[1];
    next(s[3]) := s[3];
    ...

    -- next role state
    next(Ar[0]) := (s[0] & Br[0]) | (s[1] & Xu[0] & Cr[0]);
    next(Ar[1]) := (s[0] & Br[1]) | (s[1] & Xu[1] & Cr[1]);
    next(Ar[2]) := (s[0] & Br[2]) | (s[1] & Xu[2] & Cr[2]);
    next(Ar[3]) := (s[0] & Br[3]) | (s[1] & Xu[3] & Cr[3]);
    next(Br[0]) := (s[2] & ((Xu[0] & Ds[0]) | (Xu[1] & Es[0]) |
                            (Xu[2] & Fs[0]) | (Xu[3] & Gs[0])));
    next(Br[1]) := (s[2] & ((Xu[0] & Ds[1]) | (Xu[1] & Es[1]) |
                            (Xu[2] & Fs[1]) | (Xu[3] & Gs[1])));
    next(Br[2]) := (s[2] & ((Xu[0] & Ds[2]) | (Xu[1] & Es[2]) |
                            (Xu[2] & Fs[2]) | (Xu[3] & Gs[2])));
    next(Br[3]) := (s[2] & ((Xu[0] & Ds[3]) | (Xu[1] & Es[3]) |
                            (Xu[2] & Fs[3]) | (Xu[3] & Gs[3])));
    next(Cr[0]) := (s[3]);
    next(Cr[1]) := (s[4]);
    next(Cr[2]) := (s[5]);
    next(Cr[3]) := (s[6]);
    ...

    -- next fixpoint state
    next(fixpoint) := (next(Ar[0]) = Ar[0]) &
                      (next(Ar[1]) = Ar[1]) &
                      (next(Ar[2]) = Ar[2]) &
                      ...
                      (next(Xu[2]) = Xu[2]) &
                      (next(Xu[3]) = Xu[3]);
}
```

Figure 13: Example of simulating role membership evaluation

```
Ar[0] := (s[0] & Br[0]) | (s[1] & Xu[0] & Cr[0]);
Ar[1] := (s[0] & Br[1]) | (s[1] & Xu[1] & Cr[1]);
Ar[2] := (s[0] & Br[2]) | (s[1] & Xu[2] & Cr[2]);
Ar[3] := (s[0] & Br[3]) | (s[1] & Xu[3] & Cr[3]);
Br[0] := (s[2] & ((Xu[0] & Ds[0]) | (Xu[1] & Es[0]) |
                  (Xu[2] & Fs[0]) | (Xu[3] & Gs[0])));
Br[1] := (s[2] & ((Xu[0] & Ds[1]) | (Xu[1] & Es[1]) |
                  (Xu[2] & Fs[1]) | (Xu[3] & Gs[1])));
Br[2] := (s[2] & ((Xu[0] & Ds[2]) | (Xu[1] & Es[2]) |
                  (Xu[2] & Fs[2]) | (Xu[3] & Gs[2])));
Br[3] := (s[2] & ((Xu[0] & Ds[3]) | (Xu[1] & Es[3]) |
                  (Xu[2] & Fs[3]) | (Xu[3] & Gs[3])));
Cr[0] := (s[3]);
Cr[1] := (s[4]);
Cr[2] := (s[5]);
Cr[3] := (s[6]);
...
```

Figure 14: Derived variables

We define derived variables by assigning a value to a variable without using $init$ or $next$. These assignments have the form $\lambda[j] := exp$ where exp is constructed according to Figure 12 in the same manner as presented in Section 5.1.3. Figure 14 presents the definitions of these derived variables in the case of our running example.

5.3 Output

Once we translate an RCPI into the SMV model and its specification, the SMV model checker is utilized to determine whether or not the model satisfies the specification. In the case that it does, SMV returns property verifies. In the case that the model fails to satisfy the specification, SMV will produce a counterexample. The counterexample is given by a sequence of states that falsify the specification. It is provided in the output in the form of a list of the relevant variables and their values. Variable s is a bit vector that indicates which RT statements are present in the current policy state. The value of $s[i]$ is $true$ just in case the policy state includes the statement indexed by i. The membership of roles can also be obtained: $\lambda[j]$ is $true$ when principal j is in the membership of the role λ.

The counterexample received from SMV in this encoding corresponds to $\mathcal{P}''$ in Theorem 19. As discussed in and below Proposition 20, we can use it to construct a counterexample $\mathcal{P}'$ to the original RCPI, π, by adding to it the statements of $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}-\mathsf{DefRoles}'(\pi)}$. For the purposes of presenting this counterexample, we propose using the same policy visualization techniques as used throughout this paper, augmented by highlighting the roles to which e—the witness that $[\![X.u]\!]_{\mathcal{P}'} \not\supseteq [\![A.r]\!]_{\mathcal{P}'}$—belongs. This will enable the user to see how e can be added to $[\![A.r]\!]_{\mathcal{P}'}$ without being added to $[\![X.u]\!]_{\mathcal{P}'}$.

6 Abstraction for Semi-Decision Procedures

The sound and complete decision procedure given in Section 4 constructs an AFSM having a number of reachable states that is double exponential in the size of SigRoles. As the size of SigRoles grows, the model rapidly becomes too large to be verified by a model checker. (In the formal methods literature, this is often

called the space explosion problem.) To alleviate this problem, we introduce two semi-decision procedures that can be applied to dramatically reduce the number of the states that a model checker must consider. These techniques often yield problem instances for which model-checking is an effective analysis technique where other techniques do not. The price of this effectiveness is either false negatives or false positives, depending on the semi-decision procedure uses. We provide one semi-decision procedure for which positive answers can be trusted and one for which negative answers can be trusted. Unfortunately, there are some problem instances for which neither semi-decision procedure produces an answer that can be trusted (otherwise, we could construct an efficient full-decision procedure).

The first abstraction, called *Principal Abstraction*, modifies the construction of $\mathcal{N}'$ to introduce fewer new principals than is known to be sufficient to guarantee all counterexamples are found. In other words, some counterexamples may be missed, leading to the impression that the query is satisfied when it is not (false positives). This abstraction is introduced during the construction of the AFSM, making it applicable only to model checking and, perhaps, similar techniques based on finite state machines. The second abstraction, called *Restriction Relaxation*, relaxes the restriction rule $\mathcal{R}$ in the sense that fewer roles are prevented from having statements that define them added or removed. This has the effect of making more policy states reachable, although when used in combination with the other reductions, it typically increases only the number of different role memberships (semantics) that are induced, while actually visiting and evaluating fewer policy states. In particular, the effectiveness of the COI reduction (see Section 3.1) is significantly enhanced by reducing the number of roles that are growth restricted.

The two abstractions are complementary in nature. When the Principal Abstraction produces a counterexample, it is known that the query in the RCPI is genuinely not satisfied. When the Restriction Relaxation produces an affirmative verification, it is guaranteed that the original RCPI is satisfied. Though, these techniques cannot always prove or refute the satisfiability of a given RCPI, their integration with model checking enables security experts to automatically detect errors during early policy design, which tends to be error-prone.

6.1 Principal Abstraction

The construction of the $\mathcal{N}'$ introduces an exponential number of new principals. This guarantees that a counterexample produced by the RCPI exists if and only if a counterexample can be produced by the AFSM. In many cases this number of new principals may be excessive compared to the actual number of principals necessary to expose these counterexamples. It is often the case that we can expose counterexamples with far fewer new principals, which is advantageous since we may not require as many new simple member statements and thus significantly reduce the cost of analysis. The Principal Abstraction technique reduces the number of inspected policy states by using fewer new principals than sufficient to guarantee the reduced policy model can be model checked. This technique is valuable in the sense that in the case that a counterexample is detected, it is a counterexample to the original RCPI. The technique is complete in the sense that every RCPI that is in fact satisfied is reported by the analysis to be satisfied. However, the techniques is not sound; when the analysis reports that no counterexample is detected (*i.e.*, an affirmative answer of the query is given), then we cannot be sure whether or not the RCPI is satisfied. The following theorem states the completeness of the technique. It does this by stating that if there is a counterexample that uses some number of new principals n, then there is a counterexample that uses $2^{|SigRoles(\pi)|}$ new principals. The theorem makes use of a variant of $\mathcal{N}'(\pi)$, $\mathcal{N}''(\pi, n)$, the definition of which is identical to that of $\mathcal{N}'$ except that $\mathsf{NewPrinc}(\pi)$ is replaced by $\mathsf{NewPrinc}'(\pi, n)$, which is a set of new principals, disjoint from $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{Principals}(\mathcal{R})$, and whose cardinality is n. Thus, $\mathcal{N}'(\pi) = \mathcal{N}''(\pi, 2^{|SigRoles(\pi)|})$ up to principal renaming.

Theorem 23 *Given an RCPI, $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, and a natural number n, if there exists $\mathcal{P}'$ and $E \in \mathsf{Principals}(\mathcal{P}')$ such that $E \in [\![A.r]\!]_{\mathcal{P}'}$, $E \notin [\![X.u]\!]_{\mathcal{P}'}$, and and $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R} \cap \mathsf{DefRoles}'(\pi)} \subseteq \mathcal{P}' \subseteq (\mathcal{P} \cup \mathcal{N}''(\pi, n))\restriction_{\mathsf{DefRoles}'(\pi)}$ then there exists a $\mathcal{P}''$ such that $\mathcal{P} \overset{*}{\mapsto}_\mathcal{R} \mathcal{P}''$ and there exists a principal E such that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$.*

Proof. Follows from theorem 19. ∎

Choosing an appropriate number of principals to use can be challenging. Through our experience and experimentation, we suggest that in many cases, the number of new principals should be proportional to the number of linked role expressions. Recall that a linked role expression has the form $B.r_1.r_2$ as part of a linked inclusion statement. we recommend the number of new principals to be the number of elements in $\{B.r_1.r_2 \mid \lambda \leftarrow B.r_1.r_2 \in \mathcal{P}\}$ plus one. This allows each linked role expression to produce one new sub-linked role with one new principal into the policy. Furthermore, the additional principal may serve as the witness to the counterexample. In other words, E would be the witness in the state $\mathcal{P}'$ where $E \in [\![A.r]\!]_{\mathcal{P}'}$ and $E \notin [\![X.u]\!]_{\mathcal{P}'}$. Thus given any policy with the statement $\lambda \leftarrow B.r_1.r_2$, we know that one new sub-linked role for a given linked-role expression is sufficient because new principals cannot be in $\mathsf{Principals}(\mathcal{R})$ and thus if a linked-role expression were to have a new sub-linked role, it would have to be the case that this sub-linked roles is not in the restriction rule. In other words, one new principal $C \in B.r_1$ produces the sub-linked role $C.r_2$ which must be unrestricted. Furthermore, at most one new and unrestricted sub-linked role is necessary since any additional could not introduce any principal into the membership of λ that could not have been introduced with a single unrestricted sub-linked role. In other words, since $C.r_2$ is unrestricted, introducing $D \in B.r_1$ to produce $D.r_2$ does not contribute anything to λ that could not have been contributed through $C.r_2$. Thus we consider D to be extraneous and serves only to increase the cost of analysis. It is significant to note that while we suggest a bound for the number of new principals, it is often the case that two or three principals works sufficiently well to find many counterexamples.

Figure 15 demonstrates how Principal Abstraction can be utilized to find a counterexamples by the use of three principals. The state space is significantly smaller more than the use of 2^6 or 64 new principals, which yield an RCPI that are prohibitive for model checking. In this example, it is necessary to add three new principals in order to expose a counterexample. When we apply our technique and limit the number of new principals to two, we see that no counterexample can be detected. As we have demonstrated in this example and Principal Abstraction can be an effective means of identifying a counterexample of the original policy by using a dramatically smaller policy.

6.2 Restriction Relaxation

Our second abstraction technique relaxes the restriction rule by removing one or more roles from both $\mathcal{G}_\mathcal{R}$ and $\mathcal{S}_\mathcal{R}$. Recall that the COI removes statements that define roles that are not restricted, yielding an RCPI that is equivalent with respect to analysis results, but typically smaller. Removing roles from the restriction rule can enable COI to make the RCPI smaller still—potentially much smaller.

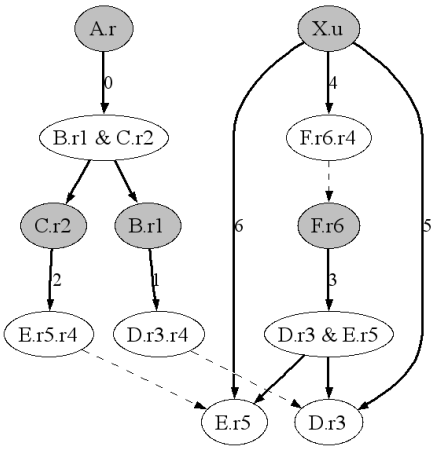
Definition 24 *Given an RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, and a set of roles $\mathcal{M}$ we define $RR(\pi, \mathcal{M}) = \langle \mathcal{P}, \mathcal{R}', X.u \sqsupseteq A.r \rangle$ in which $\mathcal{R}' = \langle \mathcal{G}_\mathcal{R} - \mathcal{M}, \mathcal{S}_\mathcal{R} - \mathcal{M} \rangle$.*

The following theorem says that the abstraction RR is sound.

Theorem 25 *Let π be any RCPI, $\mathcal{M}$ be any set of roles, and $\pi' = RR(\pi, \mathcal{M})$. If π' is satisfied, then π is also satisfied.*

Index	Policy Statement of $\mathcal{P}$
0	$A.r \leftarrow B.r_1 \cap C.r_2$
1	$B.r_1 \leftarrow D.r_3.r_4$
2	$C.r_2 \leftarrow E.r_5.r_4$
3	$F.r_6 \leftarrow D.r_3 \cap E.r_5$
4	$X.u \leftarrow F.r_6.r_4$
5	$X.u \leftarrow D.r_3$
6	$X.u \leftarrow E.r_5$

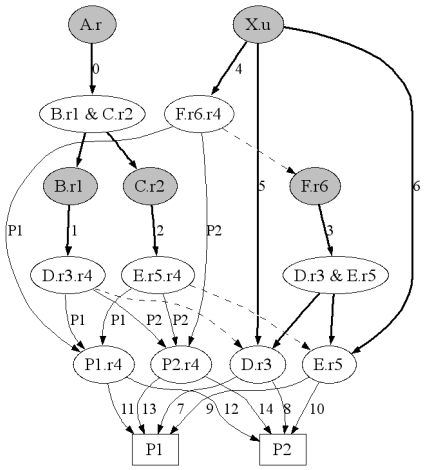
$\mathcal{G}_\mathcal{R} = \{A.r, B.r_1, C.r_2, F.r_6, X.u\}$
$\mathcal{S}_\mathcal{R} = \{A.r, B.r_1, C.r_2, F.r_6, X.u\}$



a. $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$

New Principals:
P_1, P_2

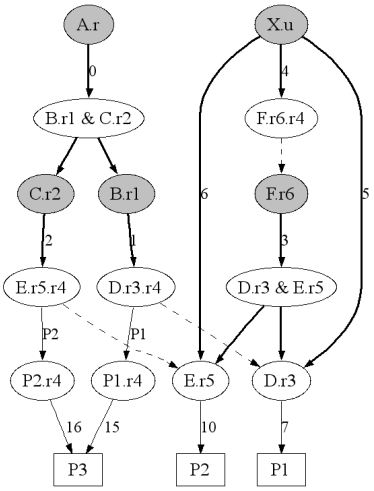
New Model Statements			
7	$D.r_3 \leftarrow P_1$	11	$P_1.r_4 \leftarrow P_1$
8	$D.r_3 \leftarrow P_2$	12	$P_1.r_4 \leftarrow P_2$
9	$E.r_5 \leftarrow P_1$	13	$P_2.r_4 \leftarrow P_1$
10	$E.r_5 \leftarrow P_2$	14	$P_2.r_4 \leftarrow P_2$



b. MaxTBE(π) w/ 2 new principals

New Principals:
P_1, P_2, P_3

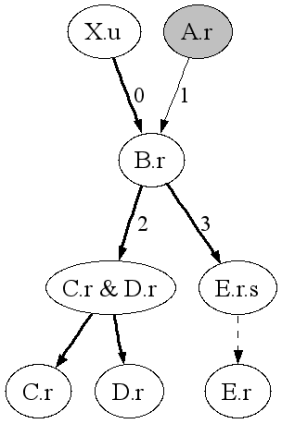
Relevant Statements	
7	$D.r_3 \leftarrow P_1$
10	$E.r_5 \leftarrow P_2$
15	$P_1.r_4 \leftarrow P_3$
16	$P_2.r_4 \leftarrow P_3$



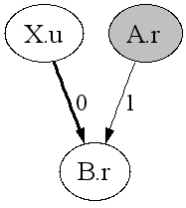
c. Counterexample of MaxTBE(π) w/ 3 new principals

Figure 15: Principal abstraction examples

Index	Policy Statement of $\mathcal{P}$
0	$X.u \leftarrow B.r$
1	$A.r \leftarrow B.r$
2	$B.r \leftarrow C.r \cap D.r$
3	$B.r \leftarrow E.r.s$
$\mathcal{G}_\mathcal{R} = \{A.r\}, \mathcal{S}_\mathcal{R} = \{X.u, B.r\}$	
RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$	
$\mathcal{S}'_\mathcal{R} = \{X.u\}$	
Policy State	Statement Indices
$\mathcal{P}$	0, 1, 2, 3
$RR\,(\mathcal{P}, \mathcal{R})$	0, 1



a. $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$



b. $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A.r \rangle$

Figure 16: Restriction relaxation (RR) example

Proof. Intuitively, this holds because any role membership of $X.u$ and $A.r$ induced by a policy state that is reachable under π will also be induced by some state reachable under π'. Thus if all states reachable under π' satisfy the query, then all states reachable π also satisfy the query. The result follows formally from the definition of reachability. ∎

The converse does not hold; when a counterexample exists for π', there may not be a counterexample for π. Thus, the technique is not complete.

As discussed in the introduction, even if the policy-state space that is reachable from $\mathcal{P}$ according to $\mathcal{R}'$ is larger than that reachable according to $\mathcal{R}$, less computational effort is typically required to solve $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq A.r \rangle$. This is because while more policy states are theoretically reachable, the COI reduction enables the analysis to visit fewer states that are redundant in the sense that another, often smaller state induces the same role memberships.

The simple example in Figure 16 illustrates the effect that Restriction Relaxation can have. The given $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is clearly satisfied because $X.u$ and $A.r$ are defined by statements 0 and 1, respectively, and no others. None of the pre-processing reductions from Section 3 can further reduce this RCPI. The largest policy state to be evaluated for π, $\mathsf{MaxTBE}(\pi)$, contains 4 significant roles, 16 new principals, and approximately 572 new simple member statements. Such an RCPI usually can not be model checked. Now suppose we apply Restriction Relaxation by removing $B.r$ from $\mathcal{G}_\mathcal{R}$ and $\mathcal{S}_\mathcal{R}$ and furthermore apply COI. The resulting $\pi' = \langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A.r \rangle$ would consist of 2 statements, and the $\mathsf{MaxTBE}(\pi')$ would require only one new principal and one simple member statement. Clearly the difference in evaluating 572 statements verses 3 statements is significant.

Discussion When an RCPI is too large for the sound and complete reductions presented in section 3 to enable satisfaction to be decided, the semi-decision procedures presented in this section can often decide whether that particular instance is satisfied. By using Restriction Relaxation, many can be determined to be satisfied. By using Principal Abstraction, many RCPIs can be determined not to be satisfied. Thus, these techniques complement each other. Of course, when the RCPI obtained by using Principal Abstraction is satisfied and the one obtained by using Restriction Relaxation is not, the semi-decision procedures are unable to determine whether the RCPI is satisfied.

7 RT-SPACE

This section describes the policy analysis framework, RT Security Policy Analysis & Correction Environment (RT-SPACE) [28] and its major components. We implement the translation approaches, all of the reductions, and the semi-complete abstraction techniques. Additionally, we build a graphical user interface to ease users efforts to input RCPIs and understand the results of the formal verifications. RT-SPACE ont only serves as a proof of concept for our theoretical work, but also provides us with the means to empirically evaluate the effectiveness of these techniques.

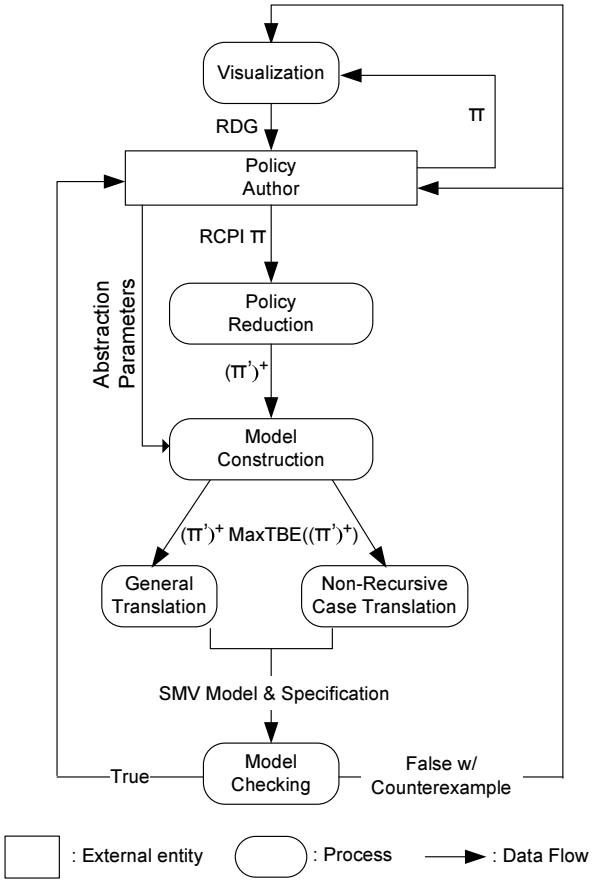


Figure 17: Policy Analysis Framework RT-SPACE

7.1 Tool Description & Usage

The RT-SPACE tool (figure 17) consists of several major components including a graphical user interface (GUI), a visualization component which is built on a graph rendering package called Grappa from AT&T Labs [3], a reduction component, a translation component, and an interface component to the Cadence SMV

model checker. This interface component includes a parser for extracting SMV results and a pretty printer component that assists in visualizing counterexample results. The tool was implemented in Java SE 1.5 and can be used independently of SMV if desired, allowing it to be fully portable since Grappa is a Java API.

The GUI provides the policy author a means of inputting RCPIs by either loading them via text files or by typing the policy, restriction rule and query directly into the user interface. In either case, the tool constructs a RDG from the input RCPI and uses Grappa to visualize the associated RDG. Recall that the RDG is a data structure that is used for various purposes in each of the major components of the policy analysis framework. It is useful not only for analyzing role-to-role and role-to-principal relationships, but also for visually depicting these relationships. In addition, it is easy to describe the sequence of roles that permit a principal to be a member of a given role using RDG and how the policy may change.

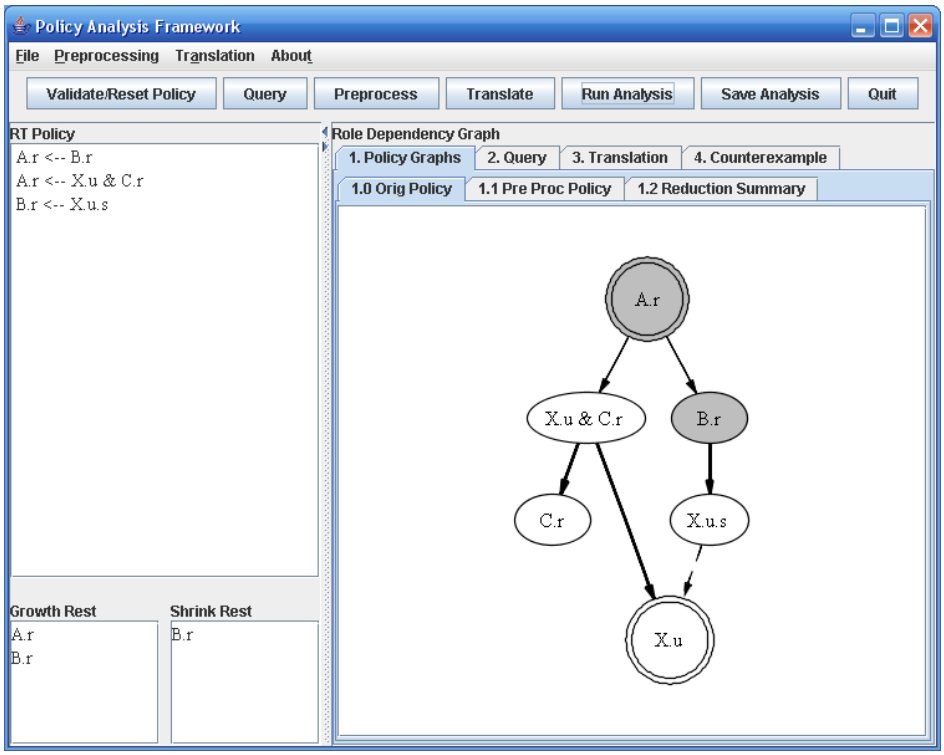
The GUI seamlessly integrates the textual description of the policy and restriction rule with the visual description of the RDG by updating the graphic with changes to the text. Figure 18 provides two screen shots of the tool. The first screen shot features the policy from the running example in Section 4. The tool's interface is composed of a text edit area on the left hand side, a visualization area on the right hand side, and a set of buttons centered above.

The usage of this tool can be described as follows. First, the policy author writes or edits a policy in the text edit area along with the restriction rule sets. The roles are expressed as `principal.roleName`, the inclusion operator is expressed as "<−−", and the intersection operator is expressed as "&". Once the policy and restriction rule are provided, the user presses the *Validate/Reset Policy* button; the tool constructs the RDG and displays it on the right hand side, as illustrated in Figure 18a. Next, the policy author provides a query by selecting the *Query* button and entering a query in a produced dialog box. For a given RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, the input query would be `X.u >> A.r`, where >> represents $\sqsupseteq$. After constructing the query, the policy author may optionally perform the set of applicable reductions by first selecting the *Preprocessing* menu and selecting some set of reductions as desired. Execution of the reductions begins after selection of the *Preprocess* button. Upon completion, the policy author may optionally select some set of abstractions [9] by selecting them from the *Translation* menu and commence translation by selecting the *Translation* button. After translation has completed, the policy author may view the SMV model and specification before continuing with the analysis. When ready, the policy author selects the *Run Analysis* button to call SMV. In the case where the model satisfies the specification, a verification message is produced along with the associated execution time. Otherwise, the model checker produces a counterexample and returns this information to the tool, which then constructs a RDG and displays it to the policy author, as illustrated in Figure 18b. The policy author may then edit the policy and restriction rule on the left hand side in pursuing a correction.

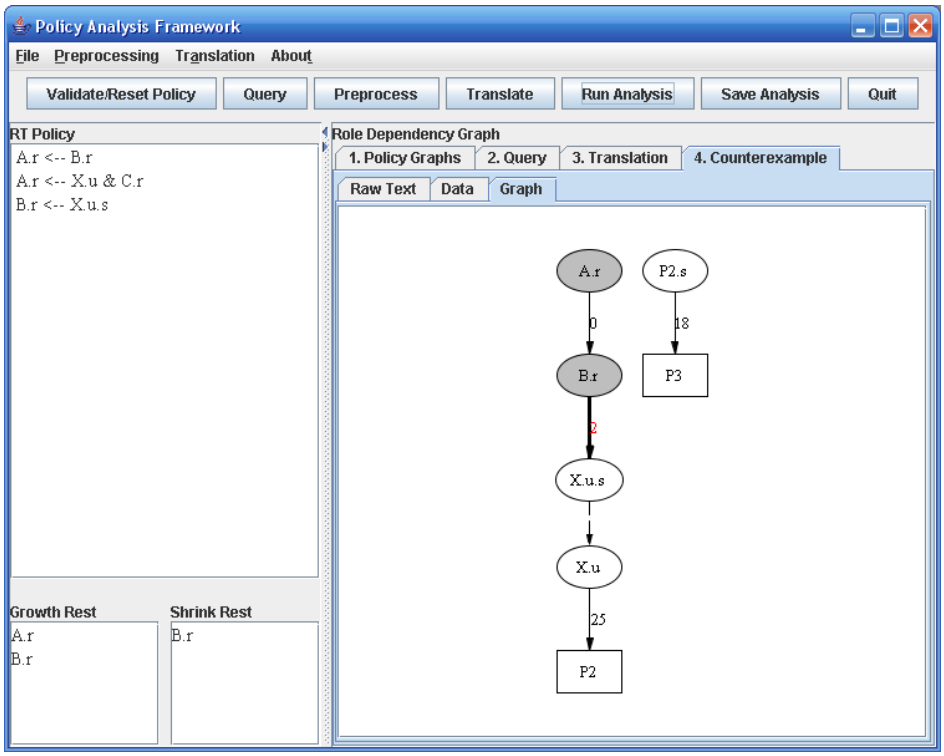
7.2 Visualization Benefits

Visualizing the RDG is beneficial because it illustrates the relationship between roles in the policy, and in particular the relationship between the queried roles. It allows us to ask some of the following questions. Is one role dependent on another role? Do the DefRoles of the queried roles overlap, and to what extent? Are there cyclic dependencies? Answering such questions is significant for two reasons. First it may yield the ability to predict the answer to the RCPI based on the structure of the graph without exploring the state space. Second, it may provide insight into effective abstraction parameters. Furthermore, it can also illustrate a counterexample to the policy author which assists in RCPI correction. By providing the counterexample in

[9] On the first attempt it is often useful to perform the translation and subsequent model checking without any abstractions in order to determine whether or not the analysis can go through. In the case that translation or model checking takes too much time, we recommend applying the abstractions as needed.



a. Input Policy



b. Counterexample

Figure 18: RT-SPACE GUI

the form of an RDG, the policy author can visualize the subgraph of roles and policy statements that permit or deny a witness principal into the queried roles. In other words, it allows the user to focus on that part of the policy in which the security property failed to be satisfied.

7.3 Using Semi-Decision Procedures

Now that we understand how to interface with the tool, let us examine how we might apply each of the abstractions for semi-decision procedures in more detail. Unlike the reductions that can be applied in every case, recall from Section 6 that the semi-decision procedures will be employed in the case that the model checker cannot finish verifying an RCPI in a reasonable time. Semi-decision procedures require some input based on the user's intuitions as to whether the given RCPI is satisfied or not. In the case that the user suspects the RCPI is satisfied, the restriction relaxations is selected; then any satisfied result produced is valid. In the case that the user suspects the RCPI is not satisfied, the principal abstraction is applied; then any counterexample produced is valid. It is important to note that if the model checker finishes, then it produces a result indicating whether or not the model satisfies the specification. If the model checker produces a result that says a principal abstracted model satisfies the specification, then we interpret this to mean that the result is inconclusive. Likewise, if the model checker produces a result that says a restriction relaxed model produces a counterexample to the specification, then we interpret this to mean that the result was again inconclusive. Therefore, between these two complementary abstractions, we may be left with an affirmation, a counterexample, or an inconclusive result.

The usage of these abstractions is straightforward. Assuming we suspect that the RCPI is not satisfied, we apply principal abstraction with a constraint that limits the analysis to consider some smaller number of new principals than being required. In our experience, two principals is often a good starting point since it allows one principal to be an independent witness and the other to be used as part of a sub-linked role in linked role expressions. If the RCPI result is affirmative, then we increment the number of new principals and re-evaluate. At some point the principal abstraction will become ineffective as the model checking fails to finish in a reasonable amount of time. We often start with principal abstraction for a newly crafted policy, since the policy is likely to falsify the property (or to produce a counterexample).

As this confidence grows, the policy author may consider reverting to the use of restriction relaxation in lieu of principal abstraction. Choosing an appropriate role to relax can seem arbitrary at first, but we suggest a few heuristics that may help. First, as mentioned in Section 6.2, try to relax a role that is defined by linked role and intersection inclusion statements as this may reduce the number of significant roles and thus the size of the reachable state space we need to consider. Second, try to relax a role defined by many simple member statements. This reduces the number of direct principals (see Section 4.1.1) as this may also reduce the size of the reachable state space. Finally, given an RDG of the RCPI, try to relax a role with a minimal number of edges from the queried roles, and if unsuccessful, attempt to relax roles progressively further away from the queried roles.

8 Empirical Findings & Analysis

This section describes the empirical evaluation conducted to determine the effectiveness of our formal analysis techniques, model checking, reductions, and abstractions. As a theoretical language, RT is not currently deployed in any operational access control system, which makes selection of naturally occurring policies problematic. In their absence, we generate test cases by fashioning by-hand RCPIs that display features of interest, as well as by randomly generating RCPIs. We evaluate these RCPIs with the following goals in

mind. First, we intend to demonstrate that despite the fact that the role-containment problem is EXPTIME complete [36], our model checking technique can in many cases produce decisive answers to problem instances in a reasonable amount of time. Second, we intend to assess the impact of our reduction techniques and provide evidence supporting our conjecture that these techniques turn a significant number of RCPIs into manageable forms. Finally, we intend to validate our semi-decision procedures and associated strategies.

We begin by describing the characteristics of problem instances, which may affect the performance and effectiveness of the analysis techniques. These features were chosen based on our experience with the role-containment problem, and provide a framework for test case selection. Next we present the partial order in which the set of reductions is applied to maximize the benefit of these techniques. Then we describe our methodology for evaluating these test cases. Finally, we conclude with an analysis of the data and summary of results.

8.1 Role Containment Problem Feature Set

In this section, we describe several features of RCPIs that often influence the cost of analysis when they are present. Our intent is to inventory features that affect cost, though we recognize our list is inevitably incomplete. In working with the problem, we have observed that RCPIs exhibiting these features tend to require greater computational resources than those that do not. These features include (1) the number of principals in the $\mathsf{MaxTBE}(\pi)$, (2) growth unrestricted roles, (3) cyclic dependencies, (4) multiply occurring linked role names, (5) the number of roles on which the membership of both queried roles depend, and (6) the number of unrestricted roles and their location in the policy in relation to the queried roles. This section discusses each of these characteristics in turn.

Number of Principals in $\mathsf{MaxTBE}(\pi)$ One characteristic that has a significant impact on the cost of analyzing a given RCPI is the number of principals in the $\mathsf{MaxTBE}(\pi)$. As the number of principals in the RCPI increases, so does the number of sub-linked roles and simple member policy statements that we need to introduce into the $\mathsf{MaxTBE}(\pi)$. Specifically, the number of new principals and direct principals influence the size of the reachable state space, in which a *direct principal* is a principal in $\{B \mid \lambda \leftarrow B \in \mathcal{P}\}$. Recall that the number of new principals is determined by the number of distinct intersections and linked roles in $\mathcal{P}$. The issue with direct principals is that the analysis must consider states that include all combinations of simple member statements introducing these principals into each growth unrestricted role. Furthermore, it may also force the consideration of many more sub-linked roles (see Section 2) than the same policy with fewer direct principals. Thus when the set of direct principals is large, the cost of analysis can be significant.

Growth Unrestricted Roles Growth unrestricted roles also play a significant part in the cost of analysis since, for each role $\lambda \notin \mathcal{G}_{\mathcal{R}}$, we need to consider all subsets of simple member statements defining λ. When the growth restricted set is small and the principal count is large, the computational effort required to evaluate an RCPI with our techniques can be prohibitive.

Cyclic Role Dependencies Cyclic role dependencies do not influence the number of reachable policy states, however they can cause the analysis to incur a significant cost in determining role membership. This cost is identified as an increase in the AFSM's state space as compared to a non-cyclic policy. It is true that the cost of this calculation is polynomial in time whereas the number of reachable states is exponential. Thus asymptotically speaking, cyclic role dependencies should not be a dominant factor in the time required for analysis. However, our technique is sensitive to this cost, due to the fact that the fixpoint calculation

it uses to compute role membership is performed by the AFSM through a sequence of state transitions. Moreover, while relatively few AFSM states are traversed in this process, the approach increases the total number of AFMS states by a factor that is exponential in the number of roles times the number of principals. In practice, these costs can often mean the difference between being able to resolve the RCPI and not.

Multiply Occurring Linked Role Names A policy feature that can influence the effectiveness of our efforts to reduce the size of the $\mathsf{MaxTBE}(\pi)$ is linked role names[10] that occur in multiple statements of $\mathcal{P}$. There are at least two cases involving linked role names that tend to make RCPIs expensive to analyze. The first case occurs when linked role names such as r_1 are found as part of defined roles such as in $F.r_1 \leftarrow e \in \mathcal{P}$. (Apropos of the last section, this may lead to cycles.) When this occurs, it ensures that F will be a principal of $\mathcal{P}$, even if it is not a direct principal of $\mathcal{P}$ (see Figure 7) and thus tends to increase the size of the $\mathsf{MaxTBE}(\pi)$.

The second case that tends to make RCPIs expensive to analyze involves linked role names shared among linked inclusion statements, such as r_1 in $\mathcal{P} = \{\lambda \leftarrow B.r.r_1, \rho \leftarrow C.r_2.r_1\}$. In this situation, it may not be safe to aggressively apply principal abstraction. Recall the example in Figure 15 from Section 6, which describes a situation where two principals are not sufficient to expose the counterexample. Analyzing a policy such as this requires at least three principals, two to construct sub-linked roles and one to act as a witness principal. Analysis of RCPIs that exhibit multiply occurring linked role names may not be able to leverage some of the abstractions for semi-decision procedures as much as analysis of those RCPIs that do not.

Number of Roles on which Queried Roles Both Depend A feature that affects the effectiveness of the COI reduction is the the number of roles on which both queried-role memberships depend. When the size of the intersection $\mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R}) \cap \mathsf{DefRoles}(\mathcal{P}, A.r, \mathcal{G}_\mathcal{R})$ is small, the COI reduction is frequently quite effective; however when the size is large, it may not be, and semi-decision procedures will often need to be tried.

Number and Location of Unrestricted Roles The number of unrestricted roles is another significant feature due to the potential for pruning the RCPI by applying the COI reduction. Given an RDG of the RCPI, the COI reduction is particularly effective when the unrestricted roles occur at a short distance from the queried roles, since the graph can be truncated at those points. In addition, the $Decompose$ reduction is influenced by the number of unrestricted roles because it requires that $A.r$ (and ideally the roles defining $A.r$) be both growth and shrink restricted before it can be applied. As mentioned in Section 3, applying COI can facilitate applying $Decompose$ because it can add these roles to $\mathcal{G}_\mathcal{R}$, the set of shrink-restricted roles. Thus, the interplay between growth and shrink restrictions is particularly consequential in the vicinity of the queried roles.

This may suggest that COI and $Decompose$ are complementary reductions since one relies on the existence of unrestricted roles while the other relies on fully restricted (growth and shrink restricted) roles. Furthermore, it may be the case that the most difficult problems are those whose roles are exclusively growth or exclusively shrink restricted, in which case neither COI nor $Decompose$ would be applicable.

[10]Recall that a linked role name is simply a role name in the set $\{r_1 \mid \lambda \leftarrow B.r.r_1 \in \mathcal{P}\}$.

8.2 Ordering of Reduction Application

The reductions described in section 3 take as input an RCPI and produce an equivalent set of one or more RCPIs. It follows from this fact that the output of one reduction may be the input to another. We now describe and demonstrate the partial order in which these reductions should be applied.

The $Decompose$ reduction produces a set of one or more RCPIs, each possibly with a new role containment query. It does not remove any statements from $\mathcal{P}$, but it may add a single statement of the form $A'.r' \leftarrow e$ to $\mathcal{P}$ if $A.r \leftarrow e \in \mathcal{P}$. This reduction relies on the COI reduction to actually remove statements, so COI should be applied after $Decompose$. Interestingly, the COI reduction may also open opportunities to apply $Decompose$ since a side effect of COI is the addition of roles to the restriction rule, thus potentially creating a situation where some role $B.r \in \mathcal{G}_\mathcal{R} \cap \mathcal{S}_\mathcal{R}$ and thus eligible as a decomposition point.

The COI reduction removes any statement that fails to define a role in $\mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P})) \cup \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P}))$. Intuitively, COI performs a type of reachability analysis on the RDG and removes any statement that defines a role from which a path to the queried roles cannot be demonstrated. Since other reduction can remove statements and break previously established paths, it is generally a good idea to repeat the COI reduction in the case that a reduction successfully removes a statement. In other words, the removal of a statement may cause the set of DefRoles to shrink, yielding opportunities to reapply COI.

The ERR removes all statements defining roles that are empty in all reachable states. Clearly such roles must be growth restricted, so any reduction that modifies $\mathcal{G}_\mathcal{R}$ must occur before ERR. Both the COI and $Decompose$ add one or more roles to $\mathcal{G}_\mathcal{R}$, and thus should be applied before ERR. The other reductions neither add statements to $\mathcal{P}$ nor add roles to $\mathcal{G}_\mathcal{R}$ and thus are independent.

Due to the multiple dependencies of each reduction, we suggest applying $Decompose$, COI, and ERR repeatedly until a fixpoint with respect to the RCPI is reached.

8.3 Empirical Evaluation

Now that we have examined some of the features that make policy analysis costly, we assess our policy analysis framework against various sample RCPIs that may exhibit one or more of these features. This exercise is useful because it helps establish a boundary between those RCPIs that can be analyzed using this technique and those for which the computational cost is prohibitive. It also helps us determine the effectiveness of the reductions. We begin with a discussion on how the test cases were chosen or generated, followed by a description of the methodology used in evaluating the test cases, an analytical discussion of the data and finally a summary of the findings.

8.3.1 RCPI Test Cases & Generation

We developed a collection of RCPIs, each designed by hand or assembled by a pseudo-random RCPI generator. In the case of manual creation, we designed policies that include the policy features discussed above. In an effort to obtain more complete coverage, we also included several RCPIs constructed automatically by a RCPI generator.

Our RCPI generator is an integrated component of our tool and as such allows for a seamless transition into the analysis tool. It uses pseudo-randomness to produce test cases that satisfy constraints specified by several input parameters; certain features such as cycles are not controlled directly by parameters. The random RCPI generator produces a single RCPI based on nine input parameters: (1) the number of statements, (2) number of roles, (3) number of principals, (4) number of role names, (5) percentage of simple

inclusion statements, (6) percentage of linked inclusion statements, (7) percentage of intersection inclusion statements, (8) percentage of growth restricted roles, and (9) percentage of shrink restricted roles. Assuming the constraints implied by these parameter values are mutually consistent (for example, not asking for a role given no principals), the algorithm generates an RCPI as follows. We first construct a set of roles and principals. Principals have the form Pc_i in which $i \in \{0..n-1\}$ where n is the maximum number of principals given by parameter 3. Role names have the form r_j in which $j \in \{0..m-1\}$ and m is the maximum number of role names given by parameter 4. We construct a set of roles by selecting principals and role names at random. Statements are then constructed at random in proportions given by input parameters 1, 5, 6, and 7. The sets of growth and shrink restricted roles are independently and randomly chosen according to parameters 8 and 9. The query is constructed by randomly choosing two of the roles.

Cyclic role dependencies arise randomly in the policy that is generated. The user can control the probability of cycles occurring by controlling the relationship between the parameter values specifying the number of principals, roles, and statements to be generated.

8.3.2 Methodology

We now describe the methodology used to evaluate policies in our test cases. The purpose of the evaluation is to determine the effectiveness of our model checking technique, reductions, and two semi-decision procedures. Effectiveness is defined as whether or not a conclusive answer to an RCPI can be determined. Recall from Section 4.1 that the size of the $\mathsf{MaxTBE}(\pi)$ and corresponding reachable state space after a reduction is applied may remain too large for model checking to produce an answer in a reasonable amount of time. Also recall from Section 7.3 that abstractions may also lead to inconclusive results in the sense that validation of the query is not conclusive when principal abstraction is used and a counterexample arising under restriction relaxation may not be reachable under the original restriction rule.

We include efficiency data as an interesting artifact of the evaluation, but is not a dominate concern due to the nature of model checking. Although it is conceivable that the model checker would run for several hours and eventually produce an answer, we find that in practice, if a model checker does not return with an answer in approximately 15 minutes and the memory usage has been near capacity for several minutes, then it is generally destined to crash due to memory exhaustion.

All test runs were produced on a 2.8 GHz Dual Core Pentium running Windows XP Service Pack 3. This machine was equipped with 4 GB of memory, although only 3.5 GB was available for applications. For a given RCPI, we record the input characteristics such as the number of each type of statement, the numbers of principals, role names, roles, significant roles, and number of growth unrestricted roles, as well as whether cycles are present in the policy. The output from the tool indicates the time to perform reductions as well as the translation time. In the case that the model checker produces an answer, SMV provides the model checking time, the number of BDD nodes used, and the number of reachable states. The number of BDD nodes demonstrates the relative memory usage between evaluations of the same RCPI with various reductions and abstractions. It is also significant to note that we enabled the BDD variable sifting option within SMV in order to better manage the size of the BDD nodes. This option uses heuristics to re-order BDD variables for the purpose of collapsing the structure into a smaller representation. It expends CPU time to reduce memory usage, which is the limiting factor in our application; the salient question is whether the model checker can complete at all, rather than how long it takes. Thus, all of our experiments were performed with this option selected.

8.3.3 Observations & Analysis

We now present the data from our evaluations. We evaluated 14 sample policies, 7 of which were presented in previous chapters as examples, and 2 of which were randomly generated. The results of these experiments can be found in Tables 1 and 2. Each test case consists of an RCPI $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, and several line items associated with it. The first line for each test case indicates the results of running the analyzer with none of our reduction or abstraction techniques applied. The following lines indicate results when reductions are performed. Recall that the set of reductions is repeatedly applied until a fixpoint is reached. Furthermore, when $Decompose$ is successful, it yields a collection of RCPIs. Each of these occupies a separate line in the table, indicated by the label Red$i$ (reduction), in which the i ranges between 1 and the number of subproblems generated by $Decompose$. The label NC is included when the non-cyclic translation was used. When reductions alone failed to enable the model checker to provide an answer, abstractions for semi-decision procedures were then employed, and the tables contain additional lines, which are labeled RR when restriction relaxation was used and PAn when principal abstraction was used with n new principals in lieu of the full number sufficient to guarantee any existing counterexamples would be found.

In addition to the columns that present characteristics of the RCPI identified in the previous section, the tables contain two others. The column labeled *Result* contains *CE* when a counterexample was found and *SAT* when the query was shown to be satisfied. It is empty if the model checker was unable to terminate normally and provide an answer. This occurred in many cases due to memory exhaustion, other model-checker resource constraints are exceeded, or an unhandled exception. The column labeled *Correct* contains *crash* when the model checker failed to provide an answer. It contains *yes* when the model checker returned the correct answer for the (sub)RCPI in question. It contains *no* when the answer found is incorrect, which reflects that an abstraction technique was used too aggressively.

We begin by making several general observations regarding this data. First, most of the baseline initial RCPI for the test cases was too large to evaluate using our general translation without any reductions or abstractions. On one hand, this was intentional since we desire to show how our reductions and abstractions allow an otherwise expensive RCPI to be evaluated, but on the other hand, many of these initial RCPI do not consist of a large set of policy statements. This reflects the difficulty of evaluating the role containment problem in general, even in small cases. Second, when abstractions are not used, our reductions and translation always produced the correct answer and in most cases required an acceptable amount of time. Recall that reductions are performed in polynomial time. On the other hand, translation can require creating an $\mathsf{MaxTBE}(\pi)$ that has size exponential in that of the original RCPI, and hence can require exponential time. Observe that some test cases exhibited reduction or translation times that seem to be zero; this is due to coarse resolution of the clock. Finally, the model checker was generally either able to provide an answer within 2 minutes or it eventually crashed.

Let us now examine a few specific test cases. In the first test case, we see that as few as 4 significant roles, coupled with a relatively large number of growth unrestricted roles, can produce a very large $\mathsf{MaxTBE}(\pi)$, despite that the number of policy statements is not large. The cost of generating the $\mathsf{MaxTBE}(\pi)$ is reflected in the time required for its translation. Unsurprisingly, we are unable to answer this RCPI. However, the reductions produced two sub-problems, each of could be handled quickly by the model checker. This improvement conforms with our observation that COI is most effective when the sets of roles on which the queried roles depend have a relatively small intersection, which is true in this case.

The second test case, borrowed from Li *et al.* [21], is known to be satisfied and it exhibits a cycle. Upon application of the reductions, we see that the problem is decomposed into a non-cyclic sub-problem and a cyclic sub-problem. The latter caused SMV to crash, so we turned to the abstraction of restriction relaxation. By removing $SA.delegatedAccess$ from the set of growth restricted roles, we were able to use restriction

Table 1: Evaluation data 1

Test Case	Policy	#	Analysis Techniques	# Stms	Princ	# Role Names	# Roles	# Sig Roles	% Inter DefRoles	% G_R	% S_R	Cyclic Depend	# AFSM States	# Policy States	Translation Time (sec)	Verificaiton Time (sec)	# BDD Nodes	# Rch States	Result	Correct
1	A.r <-- C.r B.r <-- D.r B.r <-- E.r C.r <-- D.r	1.1 1.2 1.3	Original RCPI Red1, NR Red2, NR	8 3 4	9 4 5	3 2 4	9 4 5	4 1 2	22% 22% 22%	22% 22% 22%	33% 33% 33%	no no no	2^173 2^4 2^20	2^372 2^4 2^20	0.406 0.016 0	0.015625 0.015625	467 467	1.05E+06 1.05E+06	SAT CE	crash yes yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, C.r} S_R = {B.r, D.r, X.u}																	
2	Alice.access <-- Bob HR.employee <-- HR.manager HR.manager <-- Alice HR.employee <-- HR.programmer HR.programmer <-- Bob SA.access <-- SA.manager HR.programmer <-- Carl SA.manager <-- HR.manager SA.access <-- SA.delegatedAccess & HR.employee SA.delegatedAccess <-- SA.manager.access	2.1 2.2 2.3 2.4 2.5	Original RCPI Red1, NR Red2, NR RR, Red1, NR RR, Red2, NR	10 2 8 2 4	5 2 4 2 4	5 2 6 2 5	7 2 7 2 5	3 1 3 1 2	42% 50% 43% 50% 60%	57% 50% 71% 50% 40%	71% 100% 86% 100% 60%	yes no yes no no	2^360 2^2 2^270 2^2 2^14	2^155 2^2 2^109 2^2 2^14	0.14 0 0.063 0 0	0.015625 0.015625 0.015625 0.015625	18 18 104	8 8 32768	SAT SAT SAT	crash yes crash yes yes
	Notes 1. HR.employee contains SA.access evaluates to true 2. Relaxed SA.delegatedAccess		**Restrictions** G_R = {HR.employee, SA.access, SA.delegatedAccess, SA.manager} S_R = {HR.employee,HR.manager,SA.access,SA.delgatedAccess,SA.manager}																	
3	A.r <-- B.r A.r <-- C.r A.r <-- D.r & E.r B.r <-- H.r C.r <-- F.r D.r <-- K.r	F.r <-- K.r F.r <-- Z.r.s H.r <-- F.s.t H.r <-- J.r & Y.r K.r <-- Y.r X.u <-- B.r	Original RCPI Red1, NR Red2, NR Red3, NR Red4, NR Red5, NR	13 0 0 2 4 4	13 2 2 4 6 6	4 2 2 4 5 4	14 2 2 4 6 6	7 1 1 3 3 4	36% 0% 0% 0% 17% 83%	29% 50% 50% 60% 67% 17%	29% 50% 50% 60% 67% 50%	no no no no no no	2^2 2^2 2^16 2^171 2^336	2^2 2^2 2^16 2^171 2^336	0 0 0 0.015 0.078	0.015625 0.015625 0.03125 55.2656 54.4844	17 17 211 703112 14599	4 4 65536 2.99E+51 1.40E+101	CE CE CE CE SAT	crash yes yes yes yes yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, C.r, D.r, F.r} S_R = {A.r, B.r, H.r, X.u}																	
4	Alice.friend <-- Bob.friend Bob.friend <-- Alice.friend Carol.friend <-- Dave	Alice.friend <-- Carol Bob.friend <-- Bob.friend.friend Dave.friend <-- Carol.friend	4.1 Original RCPI 4.2 Red	6 5	4 4	1 1	4 4	1 1	50% 50%	75% 75%	50% 50%	yes yes	2^56 2^56	2^19 2^19	0.031 0.015	12.6563 12.5625	3503083 3503083	3.60E+16 3.60E+16	SAT SAT	yes yes
	Notes 1. Bob.friend contains Dave.friend evaluates to true 2. Exhibits cycles		**Restrictions** G_R = {Alice.friend, Bob.friend, Dave.friend} S_R = {Alice.friend, Bob.friend}																	
5	A.r <-- C.r2 A.r <-- D.r3 A.r <-- F.r5 A.r <-- G.r6 A.r <-- H.r7 A.r <-- J.r8 A.r <-- K.r9 B.r1 <-- P1 B.r1 <-- P2 B.r1 <-- P4	C.r2 <-- P2 C.r2 <-- P3 C.r2 <-- P4 C.r2 <-- P6 D.r3 <-- P4 D.r3 <-- P5 D.r3 <-- P6 D.r3 <-- P7 E.r4 <-- B.r1 E.r4 <-- C.r2	E.r4 <-- F.r5 E.r4 <-- G.r6 E.r4 <-- H.r7 E.r4 <-- J.r8 E.r4 <-- K.r9 F.r5 <-- L.r1.r0 L.r1 <-- C.r2 & D.r3 X.u <-- E.r4 B.r1 <-- P5 E.r4 <-- D.r3	5.1 Original RCPI 5.2 RR, COI, NR	30 28	19 18	12 11	12 11	4 1	66% 73%	66% 55%	66% 82%	no no	2^1427 2^45	2^621 2^45	1.578 0.016	1.56E-02	80	1024	SAT	crash yes
			Notes 1. X.u contains A.r evaluated to true 2. Relaxed F.r5 3. Decomposes into many subproblems										**Restrictions** G_R = {A.r, B.r1, C.r2, D.r3, E.r4, F.r5, L.r1, X.u} S_R = {A.r, B.r1, C.r2, D.r3, E.r4, F.r5, L.r1, X.u}									
6	A.r <-- B.r B.r <-- C.r & D.r	B.r <-- E.r.s X.u <-- B.r	6.1 Original RCPI 6.2 Red	4 3	6 5	3 3	6 5	4 4	66% 80%	16% 20%	33% 40%	no no	2^690 2^320	2^337 2^320	0.343 0.047	49.2656	14434	2.14E+96	SAT	crash yes
	Notes 1. X.u contains A.r evaluates to true		**Restrictions** G_R = {A.r} S_R = {B.r, X.u}																	
7	A.r <-- B.r C.r <-- F.r.s E.r <-- K	A.r <-- G D.r <-- I F.r <-- L	B.r <-- C.r D.r <-- H X.u <-- B.r	C.r <-- D.r & E.r E.r <-- J X.u <-- M	7.1 Original RCPI 7.2 Red1, NR 7.3 Red2, NR	12 2 3	14 4 4	3 2 2	7 2 3	4 1 1	71% 0% 66%	28% 100% 33%	28% 100% 66%	no no no	2^1340 2^0 2^6	2^649 2^0 2^6	1.14 0.015 0	0.015625 0.03125	3 26	1 16	CE SAT	crash yes yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, E.r} S_R = {B.r, X.u}																	
8	A.r <-- B.r C.r <-- N E.r <-- P I.r <-- J.r X.u <-- C.r	A.r <-- F.r C.r <-- D.r.s F.r <-- G.r & H.r J.r <-- K.r X.u <-- K	A.r <-- L C.r <-- E.r G.r <-- Q K.r <-- S	B.r <-- M D.r <-- O H.r <-- R X.u <-- B.r	8.1 Original RCPI 8.2 Red1, NR 8.3 Red2, NR 8.4 Red3, NR	18 1 2 2	20 2 4 3	3 2 2 2	12 2 2 2	4 1 1 1	11% 0% 0% 50%	33% 50% 100% 50%	33% 100% 100% 50%	no no no no	2^1757 2^3 2^0 2^3	2^831 2^3 2^0 2^3	2.032 0 0.015 0	0.015625 0.015625 0.03125	17 3 12	4 1 4	CE CE SAT	crash yes yes yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, C.r, G.r, I.r} S_R = {E.r, F.r, J.r, X.u}																	

Notation:
Redn - Sub-RCPI #n generated from applying reductions COI, Decomposition, ERR, and WFR; **NR** - Non-Recursive version of translation; **PAn** - Principal Abstraction with n principals; **RR** - Restriction Relaxation

Table 2: Evaluation data 2

Test Case	Policy	#	Analysis Techniques	# Stms	# Princ	# Role Names	# Roles	# Sig Roles	% Inter DefRoles	% G_R	% S_R	Cyclic Depend	# AFSM States	# Policy States	Translation Time (sec)	Verificaiton Time (sec)	# BDD Nodes	# Rch States	Result	Correct
9	A.r <- B.r A.r <- Y B.r <- C.r & D.r D.r <- E.r.s	9.1 Original RCPI	E.r <- B.r X.u <-- B.r X.u <- Z	7	8	3	6	4	66%	66%	16%	yes	2^797	2^364	0.437					crash
		9.2 Red, NR		2	4	2	2	1	0%	100%	100%	no	2^0	2^0	0	0.015625	3	1	CE	yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, B.r, D.r, E.r} S_R = {X.u}																	
10	A.r <- B.r A.r <- C.r A.r <- D.r & E.r B.r <- H.r C.r <- F.r D.r <- K.r	10.1 Original RCPI	F.r <- K.r H.r <- F.s.t H.r <- J.r & Y.r K.r <- Y.r X.u <- B.r	11	11	4	12	6	41%	41%	33%	no	2^9413	2^4548	76.011					crash
		10.2 Red1, NR		0	2	2	2	1	0%	50%	100%	no	2^2	2^2	0.016	0.015625	17	4	CE	yes
		10.3 Red2, NR		3	5	2	5	3	0%	66%	100%	no	2^16	2^16	0	0.015625	211	65536	CE	yes
		10.4 Red3, NR		4	6	4	6	4	83%	16%	50%	no	2^336	2^336	0.047	54.2969	14599	1.40E+101	SAT	yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, C.r, D.r, F.r, K.r} S_R = {A.r, B.r, H.r, X.u}																	
11	A.r <- B.r A.r <- X.u & C.r B.r <- X.u.s	11.1 Original RCPI		3	4	3	4	2	25%	50%	25%	no	2^59	2^26	0.015	3.8125	135328	2.88E+17	CE	yes
		11.2 Red1, NR		1	3	3	3	2	33%	33%	33%	no	2^8	2^8	0	0.015625	51	256	SAT	yes
		11.3 Red2, NR		1	2	3	2	1	50%	50%	50%	no	2^6	2^6	0.016	0.015625	77	64	CE	yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r, B.r} S_R = {B.r}																	
12	A.r <- B.r1 & C.r2 B.r1 <-- D.r3.r4 C.r2 <-- E.r5.r4 F.r6 <-- D.r3 & E.r5	12.1 Original RCPI	X.u <-- D.r3 X.u <-- E.r5 X.u <-- F.r6.r4	7	7	8	7	6	28%	71%	71%	no	2^8769	2^4224	98.573					crash
		12.2 Red, NR		7	7	8	7	6	28%	71%	71%	no	2^4224	2^4224	4.359					crash
		12.3 Red, NR, PA2		7	7	8	7	6	28%	71%	71%	no	2^8	2^8	0.031	0.015625	93	256	SAT	no
		12.4 Red, NR, PA3		7	7	8	7	6	28%	71%	71%	no	2^15	2^15	0	0.015625	399	32768	CE	yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r., B.r1, C.r2, F.r6, X.u} S_R = {A.r, B.r1, C.r2, F.r6, X.u}																	
13	Pc4.r7 <-- Pc1 Pc4.r7 <-- Pc6.r7 & Pc5.r2 Pc5.r2 <-- Pc6.r2 Pc6.r2 <-- Pc2.r9 Pc6.r2 <-- Pc4.r7 Pc6.r2 <-- Pc5.r2.r1 Pc6.r2 <-- Pc7.r6.r5 Pc6.r7 <-- Pc2.r4 Pc7.r6 <-- Pc6.r7 Pc7.r6 <-- Pc7.r6 & Pc4.r7	13.1 Original RCPI		10	6	7	7	7	4	100%	42%	42%	yes	2^1349	2^651	1.109					crash
		13.2 Red, NR		8	6	7	7	7	4	57%	43%	57%	no	2^646	2^646	0.125					crash
		13.3 RR, Red1, NR		0	2	2	2	1	0%	50%	50%	no	2^2	2^2	0	0.03125	12	4	SAT	yes
		13.4 RR, Red2, NR		0	2	2	2	1	0%	50%	50%	no	2^2	2^2	0	0.03125	17	4	CE	yes
		13.5 RR, Red3, NR		1	2	3	2	2	0%	66%	33%	no	2^20	2^20	0	0.046875	312	1.05E+06	CE	yes
		13.6 RR, Red4, NR		2	3	4	3	2	0%	75%	50%	no	2^20	2^20	0	0.046875	312	1.05E+06	CE	yes
	Notes 1. X.u contains A.r evaluates to false		**Restrictions** G_R = {A.r., B.r1, C.r2, F.r6, X.u} S_R = {A.r, B.r1, C.r2, F.r6, X.u}																	
14	Pc1.r0 <-- Pc12.r2 Pc1.r0 <-- Pc5 Pc1.r14 <-- Pc7.r9 & Pc0.r9 Pc13.r8 <-- Pc5 Pc3.r2 <-- Pc4.r6 Pc3.r4 <-- Pc13 Pc4.r6 <-- Pc4.r6 Pc5.r5 <-- Pc5.r5 Pc7.r9 <-- Pc11.r4 & Pc1.r14 Pc7.r9 <-- Pc14 Pc7.r9 <-- Pc4.r6.r2 Pc9.r1 <-- Pc1.r14.r13 Pc9.r11 <-- Pc1.r14 & Pc5.r5 Pc9.r11 <-- Pc10 Pc9.r11 <-- Pc9.r1.r9	14.1 Original RCPI		15	12	11	13	8	0%	46%	46%	yes								crash
		14.2 Red1, NR		2	4	2	2	1	0%	100%	100%	no	2^0	2^0	0	0.015625	3	1	CE	yes
		14.3 Red2, NR		4	8	6	6	4	0%	50%	50%	no	2^399	2^399	0.078	103.313	15255	3.08E+113	CE	yes
	Notes 1. Pc13.r8 contains Pc9.r11 evaluates to false		**Restrictions** G_R = {Pc1.r0, Pc3.r4, Pc4.r6, Pc5.r5, Pc7.r9, Pc9.r11} S_R = {Pc1.r14, Pc13.r8, Pc3.r4, Pc4.r6, Pc5.r5, Pc9.r1}																	

Notation:
Redn - Sub-RCPI #n generated from applying reductions COI, Decomposition, ERR, and WFR; **NR** - Non-Recursive version of translation; **PAn** - Principal Abstraction with n principals; **RR** - Restriction Relaxation

relaxation to show that the RCPI is satisfied. Once $SA.delegatedAccess$ was made growth unrestricted, the reductions had the effect of breaking the cyclic dependence, permitting the non-recursive case translation to be used. This suggests a general strategy regarding which roles to consider first for possible restriction relaxation.

The third test case demonstrates how a difficult RCPI can be decomposed into multiple sub-problems, each of which can be evaluated independently. The original problem required the creation of an $\mathsf{MaxTBE}(\pi)$ that was so large that the translation process failed to terminate. Once the reductions were applied, the five sub-problems were small enough to be evaluated. Four of the five sub-problems yielded counterexamples, though the last sub-problem was satisfied. Thus, the initial RCPI was falsified, since there were falsified sub-problems. Another interesting aspect of this problem is that despite the fact that the reachable state space of the last sub-problem is quite large, the model checker was still able to verify that case. This supports our hypothesis that using symbolic model checking is a useful approach to the policy analysis problem.

Regarding limitations of our approach, the fourth test case demonstrates that our reductions are not always effective at significantly reducing the size of the problem that the model checker must solve. The twelfth test case illustrates the fact that applying principal abstraction too aggressively can produce incorrect results. As discussed in Section 6.1, when the set of new principals is constrained to two elements, the analysis incorrectly indicates that the query is satisfied. However when the set of new principals is enlarged to contain three principals, a counterexample is detected. Recall that this RCPI also exhibits multiply occurring linked role names, which was identified above as a characteristic that tends to indicate analysis will be more costly.

Without principal abstraction, the twelfth RCPI generates an AFSM that is much larger than the model checker can handle. While part of this is due to the number of growth unrestricted roles, the dominant factor seems to be the number of significant roles. Test cases 10 and 12 both contain 6 significant roles with approximately 2^{4200} to 2^{4550} policy states, respectively, (and many more AFSM states), and require 76 to 99 seconds to translate. On the other hand, test case 3 exhibits 7 significant roles and could not be translated, due to excessive size. This suggests that 6 is an upper bound on the number of significant roles that can be handled by our approach.

The question of where the boundary lies between those RCPI that can be analyzed and those that cannot is more complex since the size of the state space is not always directly correlated with the ability to obtain a definitive answer from SMV. The upper bound on size of the state space that SMV can handle depends heavily on the number and ordering of BDD variables. Though it is clear that the larger the state space, the less probable that SMV will reach a conclusion, these metrics do not yield a sharp cut off point.

The thirteenth test case demonstrates how restriction relaxation can be usefully applied to an RCPI even when the model checker discovers a counterexample. Recall that when no counterexample is found after restriction relaxation is performed, we are guaranteed that the original RCPI also has no counterexample. However, when a counterexample is found, if it is a policy that contains no new statements for roles that are growth restricted in the original RCPI, it constitutes a counterexample to the original RCPI as well. In the case of test case thirteen, the model checker produced counterexamples from three of the decomposed sub-problems that were also counterexamples to the original RCPI. We manually examined each of the three counterexample in the context of the original RCPI to determine that they corresponded to counterexamples in it as well.

The remaining test cases exhibit characteristics similar to those already discussed. They tend to support the hypothesis that our model checking approach can be effective, and that the reductions and abstractions we have introduced significantly enlarge the set of RCPIs for which this is so. It should be noted that while the test cases we present demonstrate effectiveness, there are many RCPIs that our approach is unable to

solve. Unfortunately, such cases are commonplace. This is to be expected, given the high complexity of the role containment problem. In large measure, our goal is to discover the limits of automated policy analysis in practice and to this end we next discuss our observations regarding the policy features that influence the difficulty of evaluating an RCPI.

The results of these experiments and our other experience working with the role containment problem tend to support the importance of certain characteristics in determining whether our techniques are effective. To summarize, our reductions are most effective in limiting the size of the $\mathsf{MaxTBE}(\pi)$ when the queried roles do not depend on a large number of growth or shrink restricted roles in the original RCPI. In particular, it is very advantageous if the number of such roles on which *both* queried roles depend is small. Our techniques are less successful when these role sets are large and have cyclic dependencies. Our experience is that it is quite difficult to effectively apply restriction relaxation when cycles appear in this part of the policy. Furthermore when these roles are defined by a large number of simple member statements, the resource requirements to analyze the RCPI can be impractical to meet. Principal abstraction only serves to limit the number of new principals. So when the original RCPI contains a large number of principals, this can result in an excessively large number of simple member statements in the $\mathsf{MaxTBE}(\pi)$. True, this problem is mitigated when a large collection of principals occupy the same set of roles, since, as we have shown in Section 2, in this case all but one representative of such a collection can be eliminated. However, when principals are added to many different sets of roles, the number of representative principals can still be very large.

9 Related Work

This section describes related work on verifying security properties of trust management and access control systems. In many cases a particular related work evaluates a verification technique for a specific security property and/or specific access control system. We compare the strengths and limitations of each related work against our framework. In addition, we examine the applicability of each work to our problem.

9.1 Comparison Criteria

We analytically compare our framework to related verification techniques within two categories. The first category includes verification techniques related to trust management systems, which includes the RT and SPKI/SDSI policy languages. These languages support delegation of permission and delegation of authority in a decentralized environment, and as such the associated verification techniques are highly relevant. The second category includes verification techniques related to centralized access control systems. The majority of verification work is associated with role-based access control (RBAC) systems, some with delegation and administrative extensions. It is significant to note that in centralized systems, the security of the system depends primarily on the actions of security administrators rather than stakeholders. This often leads to a different type of analysis than found in trust management, although it is plausible to extend this type of trust management analysis to RBAC systems. It is not clear whether current verification techniques can be applied to our trust management analysis. Furthermore, to the best of our knowledge, none of approaches in these two categories provides intensive automated reduction and abstraction techniques to reduce the computation cost of formal analysis of access control models.

The comparison criteria includes the following. First, we compare theoretical upper and lower bounds of each technique against our own. Second, we compare the implementations of their technique and counterexample generation. Finally, we compare usage processes for verifying and analyzing security policies

even in the cases where the verification is intractable.

9.2 Trust Management Verification Techniques

Li *et al.* [21] originally propose security analysis of RT within the context of Datalog. They establish theoretical upper and lower bounds for each of the policy classes in the RT family with respect to simple and bounded safety, availability, and role containment properties. For example, each of these property categories can be evaluated in polynomial time, with exception to role containment. The complexity of role containment analysis is determined by the class of the policy in question. For $RT[\,]$, the analysis is in polynomial time, whereas $RT[\leftarrow]$ is PSPACE-complete and $RT[\cap]$ is co-NP. The final class, $RT[\leftarrow, \cap]$ was demonstrated to be co-NEXPTIME. While Datalog semantics were useful to illustrate and prove many of these complexity bounds, the associated algorithms presented in their paper provide no evidence as to how or why a policy may fail to satisfy a particular property.

Sistla and Zhou [36] also provide a framework for reasoning about security analysis of RT policies. Of significant value is their proof of a tight upper and lower bound EXPTIME complexity for role containment queries in the $RT[\leftarrow, \cap]$ class. They accomplish this by transforming the analysis into a language containment problem for unbounded systems. This involves an algorithm that searches through every sequence of every principal-to-role assignment for those sequences that are consistent with the growth restrictions. If all of the consistent sequences satisfy the role containment property, then the algorithm claims the policy satisfies the property. When a role containment $\rho \sqsupseteq \lambda$ is not satisfied in all reachable policies, it is not clear that Sistla's method could be made to find such a policy; the technique only reports that one exists. The reason that it is able to improve on the complexity bound of the approach we use is precisely that it does not actually introduce new principals, which makes counterexample generation difficult. While beneficial for establishing a theoretical bound, we view this as a a serious limitation for a practical technique. Our principle goal is to provide meaningful feedback to the analysis tool user.

Jha and Reps [17] verify such properties as authorization, availability, and shared access of the SPKI/SDSI policy language through the use of push down systems and specialized algorithms. Push down systems can be used to represent language containment problems for unbounded systems. This approach is considered a type of model checking and has been shown to run in many cases in time polynomial to the size of the certificate set. SPKI/SDSI can be viewed as a subset of RT, specifically the policy class $RT[\leftarrow]$. This implies that Jha's approach does not support intersection inclusion statements. In addition, the type of analysis examined in this work assumes that all principals issuing delegation credentials are trusted, which is not the type of security analysis we examine. No mention was made regarding reduction or abstraction techniques, which are significant in many cases because otherwise the problem may not be checkable. It is doubtful that our techniques would be immediately applicable since they depend on a restriction rule which is not part of this work.

Our techniques are applicable to security analysis of Simple Distributed Security Infrastructure (SDSI) [30], as SDSI policies can be translated into RT policies almost trivially. We conjecture that our techniques may also be applicable to the analysis of other policy languages, such as Cassandra [4], that are founded on variants of Datalog. However, this question remains open.

9.3 Other Access Control Verification Techniques

Fisler *et al.* [10] analyze the impact of policy changes on RBAC systems using their own model checking tool called Margrave implemented in Scheme. Such policies are represented as multi-terminal BDD's for efficient storage and manipulation. Their work focuses on the process of validating a policy change before

committing it, which is applicable to security administrators. They verify separation of duty properties and their tool produces counterexamples when a property is not satisfied. While they do not provide any theoretical bounds on performance, they provide a concrete example with associated performance metrics.

Schaad *et al.* [34] also verify separation of duty properties in the context of a work flow process utilizing an RBAC system. The goal of this work is to identify a sequence of delegations and revocations that may place a principal into two or more roles that would be characterized as unsafe. While their version of RBAC supports delegation and revocation of permissions, it does not support delegation of authority as seen in modern trust management languages. The authors propose the use of NuSMV, which is a model checking tool in the SMV family and supports features such as bounded model checking and simulation of the reachable state space. However, it is unclear as to the computational complexity of their approach or how they might reduce or abstract the problem to address the state-explosion problem.

Hansen *et al.* [13] utilize model checker SPIN to verify various static and dynamic separation of duty properties in RBAC. Their approach does not support any form of delegation, and the authors do not provide a theoretical performance bound.

Koch *et al.* [18] reason about RBAC through a graph-based formalism. In their graph, nodes represent users, roles, sessions and permissions, while edges represent associations between the nodes. Graph transformation rules allow users, roles and permissions to be added or removed, as well as activating or deactivating sessions. Graphical constraints are used to specify consistency properties, often represented as graph of a forbidden structure of nodes and edges. The authors develop techniques for proving the correctness dynamic separation of duty specifications using this formalism. The contribution of this work is a framework for comparing different access control models and analyzing the effect of combining multiple policies. Some of the advantages of this approach include intuitive visualization of the RBAC policy as well as the application of graph theory. While we utilize graphs for visualization and analysis, there are several reasons why this approach does not address our needs. First, it is unclear how to extend the formalism to represent delegation of authority or rights. Furthermore, it does not support role intersection as an operation in their framework. Second, the analysis is not automated, although certain graph transformations are.

Stoller *et al.* [33, 38, 39] study the complexity of problems related to ours, but in the context of RBAC. They investigate user-role reachability analysis in their parameterized administrative role access control (ARBAC) system. User-role reachability asks whether some set of administrators could modify the RBAC policy so that a target user becomes a member of a specified role. One can use such queries to verify safety and availability properties. Depending on the expressive power of administrative policy, the complexity of the user-role reachability problem ranges from polynomial time to PSPACE-complete. Using a symbolic state graph, they are able to produce a path by which the a target user be made a member of the given role.

Jha *et al.* [16] reason about classes of security analysis problems related to RBAC and identify the computational complexity of such problems as PSPACE-complete. By exploring the factors that influence the complexity of a given problem, they were able to identify sub-categories of problems that can be decided in polynomial time. In addition to their theoretical results, they experimented with formal techniques including symbolic model checking and logic programming. In more than half of their experiments, preprocessing techniques were necessary in order to scope the problem to a size that could be evaluated. The type of research in this work on RBAC is the most similar to our research of practical security analysis of RT.

Zhang *et al.* [40, 41, 42] provide a model checking algorithm for evaluating access control systems and present the computational complexity of the algorithm. This approach can be used to detect, in an access control policy, errors caused by the interactions of policy rules and coalitions among multiple agents. Our goal is different from theirs in that we intend to check if a set of principals (*e.g.*users or subsystems) can access the resources no matter how policy evolves in a decentralized environment, in which authorities can

be delegated.

Other work [9, 11, 12, 26, 35] also supports the use of formal techniques and tools to verify properties of security policies, however none significantly address practical concerns such as usability, performance, and scalability. Furthermore, it is unclear as to how we might use these techniques to address role containment in a delegation language such as RT. Our techniques presented in this paper can be used for certain security analysis problems within the context of ARBAC. As shown by Li and Tripunitara [22], two security analysis problems arising in ARBAC can be reduced to security analysis problems involving RT, namely the Assignment And Trusted Users (AATU) problem and the Assignment And Revocation (AAR) problem.

10 Conclusion

Crafting and validating decentralized access control policies that reflect the author's intention is a difficult task for stakeholders. They require automated means of reasoning about the protection of their resources as the complexity and intractability of policies make manual approaches unreasonable or impossible. Ideally, the techniques and tools can also provide information on why and how their policies fail to meet their expectations.

Our contribution meets these requirements by basing our automated security analysis approach on model checking. We develop a collection of reduction techniques and semi-decision procedures. Such techniques and procedures may be able to reduce problem instances into a form that can be automatically verified. Furthermore, we include proofs demonstrating correctness of our techniques and an empirical evaluation of our techniques showing the effectiveness and performance of our approach. Our findings show that many RT policy analysis instances can be effectively verified using the methodology of associating reductions with model checking technique.

Appendix

This appendix contains proofs of the theorems in the body of the paper.

Theorem 9 (Section 3.1) *Given any RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, COI ($\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$) is satisfied if and only if $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is satisfied.*

We prove the soundness of COI in three parts corresponding to the following three lemmas. The first lemma provides a proof of soundness for the definition of $\mathcal{P}'$ as per (1). The second lemma provides a proof of soundness for the definition of $\mathcal{G}'_{\mathcal{R}}$ as per (3), and finally the third lemma provides a proof of soundness for the definition of $\mathcal{S}'_{\mathcal{R}}$ as per (4).

Lemma 26 *Given any RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$, $\langle \mathcal{P}', \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is satisfied if and only if $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is satisfied, where $\mathcal{P}'$ is defined by (1).*

Proof. We prove entailment in both directions. In the "only if" part, let $\mathcal{P}'''$ be reachable from $\mathcal{P}'$ under $\mathcal{R}$ and let E satisfy $E \notin [\![X.u]\!]_{\mathcal{P}'''} \wedge E \in [\![A.r]\!]_{\mathcal{P}'''}$. Let $\mathcal{P}'' = \mathcal{P}''' \cup \mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}}$.

We show $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$. Consider any $\lambda \leftarrow e \in \mathcal{P} - \mathcal{P}''$. We know that $\lambda \notin \mathcal{S}_{\mathcal{R}}$ because $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}''$, so removing $\lambda \leftarrow e$ is permitted by $\mathcal{R}$. Now consider any $\lambda \leftarrow e \in \mathcal{P}'' - \mathcal{P}$. By construction of $\mathcal{P}''$ it must be that $\lambda \leftarrow e \in \mathcal{P}'''$, which is reachable from $\mathcal{P}'$. Since $\mathcal{P}' \subseteq \mathcal{P}$, it follows that $\lambda \leftarrow e \in \mathcal{P}''' - \mathcal{P}'$, which implies that $\lambda \notin \mathcal{G}'_{\mathcal{R}}$. Since $\mathcal{G}_{\mathcal{R}} \subseteq \mathcal{G}'_{\mathcal{R}}$, $\lambda \leftarrow e$ can be added to $\mathcal{P}$, as required.

Next we show $E \notin [\![X.u]\!]_{\mathcal{P}''}$ and $E \in [\![A.r]\!]_{\mathcal{P}''}$. The latter follows easily from $E \in [\![A.r]\!]_{\mathcal{P}'''}$ and from $\mathcal{P}''' \subseteq \mathcal{P}''$, which tells us that $[\![A.r]\!]_{\mathcal{P}'''} \subseteq [\![A.r]\!]_{\mathcal{P}''}$. We show $E \notin [\![X.u]\!]_{\mathcal{P}''}$ by proving that for all roles $B.r' \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$, $[\![B.r']\!]_{\mathcal{P}''} \subseteq [\![B.r']\!]_{\mathcal{P}'''}$. For this we use induction on i to show that if $D \in T_{\mathcal{P}''}\uparrow^i [\![B.r']\!]$, then $D \in T_{\mathcal{P}'''}\uparrow^j [\![B.r']\!]$, for some $j \in \mathbb{N}$. The base case is trivial. For the step, consider any $D \in T_{\mathcal{P}''}\uparrow^{i+1} [\![B.r']\!]$ and fix the statement $B.r' \leftarrow e$ that is used to introduce $D \in T_{\mathcal{P}''}\uparrow^{i+1} [\![B.r']\!]$. There are four cases based on the structure of e.

Case 1: $e = D$**.** Because $B.r' \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$, it follows that $B.r' \leftarrow D \in \mathcal{P}'$. If $B.r' \in \mathcal{S}_\mathcal{R}$, then $B.r' \leftarrow D \in \mathcal{P}'''$ by definition of reachability. If not, then $B.r' \leftarrow D \in \mathcal{P}'''$ follows by construction of $\mathcal{P}''$. In either case we have $D \in T_{\mathcal{P}'''}\uparrow^1 [\![B.r']\!]$.

Case 2: $e = C.r_1$**.** We obtain $B.r' \leftarrow C.r_1 \in \mathcal{P}'''$ as we did $B.r' \leftarrow D \in \mathcal{P}'''$ in the first case. It follows that $D \in T_{\mathcal{P}''}\uparrow^i [\![C.r_1]\!]$. We show that there is a j for which $D \in T_{\mathcal{P}'''}\uparrow^j [\![C.r_1]\!]$. By definition of DefRoles, $B.r' \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$ implies either $C.r_1 \notin \mathcal{S}_\mathcal{R}$ or $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$. When $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$, we obtain $D \in T_{\mathcal{P}'''}\uparrow^j [\![C.r_1]\!]$ from the induction assumption. When $C.r_1 \notin \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$, it follows that there is no statement defining $C.r_1$ in $\mathcal{P}'$. Thus it must be that $C.r_1 \notin \mathcal{G}_\mathcal{R}$ and that $C.r_1 \leftarrow D \in \mathcal{P}'' - \mathcal{P}$. (Recall that by Theorem 4 we can assume without loss of generality that $\mathcal{P}'' - \mathcal{P}$ consists of simple member statements.) Since $C.r_1 \notin \mathcal{S}_\mathcal{R}$, it follows from the construction of $\mathcal{P}''$ that $C.r_1 \leftarrow D \in \mathcal{P}'''$. Therefore $D \in T_{\mathcal{P}'''}\uparrow^1 [\![C.r_1]\!]$. We now obtain $D \in T_{\mathcal{P}'''}\uparrow^{j+1}$ $[\![B.r']\!]$, as required.

The cases in which $e = C.r_1.r_2$ and $e = C.r_1 \cap F.r_2$ are similar to the second case above, thus completing the "only if" part of the proof. In the "if" part of the proof, we use a variant of DefRoles that includes roles in $\mathcal{M}$ as well as roles used in their definitions.

Definition 27 *Let Λ and $\mathcal{M}$ be sets of roles, and $\mathcal{P}$ be a policy. We define* $\mathsf{DefRoles}^+(\mathcal{P}, \rho, \mathcal{M})$ *to be the least set of roles $\mathcal{O}$ satisfying the following conditions:*

- $\rho \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \in \mathcal{M} \ \wedge \ \lambda \leftarrow B.r_1 \in \mathcal{P}) \Rightarrow B.r_1 \in \mathcal{O}$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \in \mathcal{M} \ \wedge \ \lambda \leftarrow B.r_1.r_2 \in \mathcal{P} \wedge D \in Principals(\mathcal{P})) \Rightarrow (B.r_1 \in \mathcal{O} \wedge D.r_2 \in \mathcal{O})$
- $(\lambda \in \mathcal{O} \ \wedge \ \lambda \in \mathcal{M} \ \wedge \ \lambda \leftarrow B.r_1 \ \cap \ C.r_2 \in \mathcal{P}) \Rightarrow (B.r_1 \in \mathcal{O} \wedge C.r_2 \in \mathcal{O})$

Let $\mathcal{P}''$ be reachable from $\mathcal{P}$ and let E satisfy $E \notin [\![X.u]\!]_{\mathcal{P}''} \wedge E \in [\![A.r]\!]_{\mathcal{P}''}$. It is convenient in this case to construct $\widehat{\mathcal{P}}$ from $\mathcal{P}''$ which is also reachable from $\mathcal{P}$ and satisfies $E \notin [\![X.u]\!]_{\widehat{\mathcal{P}}} \wedge E \in [\![A.r]\!]_{\widehat{\mathcal{P}}}$.

Let $\mathcal{P}_0 = \{F.r'' \leftarrow D \mid F.r'' \in \mathsf{DefRoles}^+(\mathcal{P}, A.r, \mathcal{G}_\mathcal{R}) - \mathcal{G}_\mathcal{R} \ \wedge \ D \in [\![F.r]\!]_{\mathcal{P}''}\}$.

Let $\widehat{\mathcal{P}} = (\mathcal{P}'' - \mathcal{P}''\restriction_{DefRoles^+(\mathcal{P},X.u,\mathcal{S}_\mathcal{R}) - (\mathcal{S}_\mathcal{R} \cup \mathsf{DefRoles}(\mathcal{P},A.r,\mathcal{G}_\mathcal{R}))}) \cup \mathcal{P}_0$. It can be shown by induction that for all $B.r' \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R})$, $[\![B.r']\!]_{\widehat{\mathcal{P}}} \subseteq [\![B.r']\!]_{\mathcal{P}''}$ and that for all $B.r' \in \mathsf{DefRoles}(\mathcal{P}, A.r, \mathcal{G}_\mathcal{R})$, $[\![B.r']\!]_{\widehat{\mathcal{P}}} = [\![B.r']\!]_{\mathcal{P}''}$. It follows from this that $E \notin [\![X.u]\!]_{\widehat{\mathcal{P}}} \wedge E \in [\![A.r]\!]_{\widehat{\mathcal{P}}}$.

Now we construct $\mathcal{P}'''$, reachable from $\mathcal{P}'$, by $\mathcal{P}''' = \widehat{\mathcal{P}} \ \restriction_{M(\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle)} \ \cup \ \mathcal{P}_0$, in which $M(\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle) = \mathsf{DefRoles}(\mathcal{P}, X.u, \mathcal{S}_\mathcal{R}) \cup \mathsf{DefRoles}(\mathcal{P}, A.r, \mathcal{G}_\mathcal{R})$. It can be shown that $\mathcal{P}'''$ is reachable from $\mathcal{P}'$. It can now be shown that $[\![B.r']\!]_{\mathcal{P}'''} \subseteq [\![B.r']\!]_{\widehat{\mathcal{P}}}$ for all $B.r'$ defined in $\mathcal{P}'''$. From this we get $E \notin [\![X.u]\!]_{\mathcal{P}'''}$.

To show that $E \in [\![A.r]\!]_{\mathcal{P}'''}$, one can use induction on i to show that for all roles $B.r' \in M(\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle)$, if $D \in T_{\widehat{\mathcal{P}}}\uparrow^i [\![B.r']\!]$ then there is a $j \in \mathbb{N}$ such that $D \in T_{\mathcal{P}'''}\uparrow^j [\![B.r']\!]$. ∎

Lemma 28 *Given any RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \ \sqsupseteq \ A.r \rangle$, $\langle \mathcal{P}, \mathcal{R}', X.u \ \sqsupseteq \ A.r \rangle$ is satisfied if and only if $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ is satisfied, where $\mathcal{R}' = (\mathcal{G}'_\mathcal{R}, \mathcal{S}_\mathcal{R})$ and $\mathcal{G}'_\mathcal{R}$ is defined by (3).*

Proof.

Recall from (3) that $\mathcal{G}'_{\mathcal{R}} = \mathcal{G}_{\mathcal{R}} \cup \Gamma$, where $\Gamma = \mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P})) - \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P}))$. We assume without loss of generality the reachable state $\mathcal{P}'$ is a subset of $\mathcal{P}$ along with simple member statements defining roles in $\mathcal{P}$. The if part is trivial because any $\mathcal{P}''$ that is reachable from $\mathcal{P}$ under $\mathcal{R}'$ is also reachable from $\mathcal{P}$ under $\mathcal{R}$. So if $\mathcal{P}''$ shows $\langle \mathcal{P}, \mathcal{R}', X.u \sqsupseteq A.r\rangle$ is not satisfied, it also shows $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r\rangle$ is not satisfied.

For the only if part, assume $\mathcal{P}'$ is reachable from $\mathcal{P}$ under $\mathcal{R}$ and that $E \in [\![A.r]\!]_{\mathcal{P}'}$ and $E \notin [\![X.u]\!]_{\mathcal{P}'}$. Construct $\mathcal{P}'' = \mathcal{P}' - \{B.r \leftarrow e \in \mathcal{P}' - \mathcal{P} \mid B.r \in \Gamma\}$. This removes all the statements the addition of which to $\mathcal{P}'$ would violate $\mathcal{G}'_{\mathcal{R}}$. Thus $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}'$.

It remains to be shown that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$. The former can be shown by a simple induction on i by proving that for all $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P}))$, and for all principals D, if $D \in T_{\mathcal{P}'} \uparrow^i [\![C.r_1]\!]$, then $D \in T_{\mathcal{P}''} \uparrow^\omega [\![C.r_1]\!]$. The step goes through easily because $\mathcal{P}' \restriction_{\mathsf{DefRoles}(\mathcal{P},A.r,\mathsf{Roles}(\mathcal{P}))} = \mathcal{P}'' \restriction_{\mathsf{DefRoles}(\mathcal{P},A.r,\mathsf{Roles}(\mathcal{P}))}$. To show $E \notin [\![X.u]\!]_{\mathcal{P}''}$ we use proof by contradiction and show that if $E \in [\![X.u]\!]_{\mathcal{P}''}$ then $E \in [\![X.u]\!]_{\mathcal{P}'}$. We prove by induction that for all $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P}))$ and all principals D, if $D \in T_{\mathcal{P}''} \uparrow^i [\![C.r_1]\!]$, then $D \in T_{\mathcal{P}'} \uparrow^\omega [\![C.r_1]\!]$. Again, the step goes through easily because $\mathcal{P}'' \restriction_{\mathsf{DefRoles}(\mathcal{P},X.u,\mathsf{Roles}(\mathcal{P}))} \subseteq \mathcal{P}' \restriction_{\mathsf{DefRoles}(\mathcal{P},X.u,\mathsf{Roles}(\mathcal{P}))}$.

■

Lemma 29 *Given any RCPI $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r\rangle$, $\langle \mathcal{P}, \mathcal{R}', X.u \sqsupseteq A.r\rangle$ is satisfied if and only if $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r\rangle$ is satisfied, where $\mathcal{R}' = (\mathcal{G}_{\mathcal{R}}, \mathcal{S}'_{\mathcal{R}})$ and $\mathcal{S}'_{\mathcal{R}}$ is defined by (4).*

Proof.

Recall from (4) that $\mathcal{S}'_{\mathcal{R}} = \mathcal{S}_{\mathcal{R}} \cup \Sigma$, where $\Sigma = \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P})) - \mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P}))$. We assume without loss of generality the reachable state $\mathcal{P}'$ is a subset of $\mathcal{P}$ along with simple member statements defining roles in $\mathcal{P}$. The if part is trivial because any $\mathcal{P}''$ that is reachable from $\mathcal{P}$ under $\mathcal{R}'$ is also reachable from $\mathcal{P}$ under $\mathcal{R}$. So if $\mathcal{P}''$ shows $\langle \mathcal{P}, \mathcal{R}', X.u \sqsupseteq A.r\rangle$ is not satisfied, it also shows $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r\rangle$ is not satisfied.

For the only if part, assume $\mathcal{P}'$ is reachable from $\mathcal{P}$ under $\mathcal{R}$ and that $E \in [\![A.r]\!]_{\mathcal{P}'}$ and $E \notin [\![X.u]\!]_{\mathcal{P}'}$. Construct $\mathcal{P}'' = \mathcal{P}' \cup \{B.r \leftarrow e \in \mathcal{P} - \mathcal{P}' \mid B.r \in \Sigma\}$. This adds all the statements the removal of which from $\mathcal{P}'$ would violate $\mathcal{S}'_{\mathcal{R}}$. Thus $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}'$.

It remains to be shown that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$. The former can be shown by a simple induction on i by proving that for all $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, A.r, \mathsf{Roles}(\mathcal{P}))$, and for all principals D, if $D \in T_{\mathcal{P}'} \uparrow^i [\![C.r_1]\!]$, then $D \in T_{\mathcal{P}''} \uparrow^\omega [\![C.r_1]\!]$. The step goes through easily because $\mathcal{P}' \restriction_{\mathsf{DefRoles}(\mathcal{P},A.r,\mathsf{Roles}(\mathcal{P}))} \subseteq \mathcal{P}'' \restriction_{\mathsf{DefRoles}(\mathcal{P},A.r,\mathsf{Roles}(\mathcal{P}))}$. To show $E \notin [\![X.u]\!]_{\mathcal{P}''}$ we use proof by contradiction and show that if $E \in [\![X.u]\!]_{\mathcal{P}''}$ then $E \in [\![X.u]\!]_{\mathcal{P}'}$. We prove by induction that for all $C.r_1 \in \mathsf{DefRoles}(\mathcal{P}, X.u, \mathsf{Roles}(\mathcal{P}))$ and all principals D, if $D \in T_{\mathcal{P}''} \uparrow^i [\![C.r_1]\!]$, then $D \in T_{\mathcal{P}'} \uparrow^\omega [\![C.r_1]\!]$. Again, the step goes through easily because $\mathcal{P}'' \restriction_{\mathsf{DefRoles}(\mathcal{P},X.u,\mathsf{Roles}(\mathcal{P}))} = \mathcal{P}' \restriction_{\mathsf{DefRoles}(\mathcal{P},X.u,\mathsf{Roles}(\mathcal{P}))}$.

■

Lemma 30 *Given a policy $\mathcal{P}$, restriction rule $\mathcal{R}$ and $B.r \in \mathsf{Roles}(\mathcal{P})$, $[\![B.r]\!]_{UB(\mathcal{P},\mathcal{R})} = \emptyset$ if and only if for every $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ satisfies $[\![B.r]\!]_{\mathcal{P}'} = \emptyset$.*

Proof. Follows immediately from Proposition 3.6 [21]. ■

Theorem 11 (Section 3.2) *Let $\mathcal{P}$ be any policy and $\mathcal{R}$ be any restriction rule. (1) For all $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and $\mathcal{P}' - \mathcal{P}$ contains type 1 statements only, there exists $\mathcal{P}''$ such that $ERR(\mathcal{P}, \mathcal{R}) \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}''$, $\mathcal{P}'' -$*

$ERR(\mathcal{P},\mathcal{R})$ contains type 1 statements only, and for all $B.r \in \mathsf{Roles}(\mathcal{P}') \cup \mathsf{Roles}(\mathcal{P}'')$, $[\![B.r]\!]_{\mathcal{P}'} = [\![B.r]\!]_{\mathcal{P}''}$. Conversely, it is also the case that (2) for all $\mathcal{P}''$ such that $ERR(\mathcal{P},\mathcal{R}) \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}''$ and $\mathcal{P}'' - ERR(\mathcal{P},\mathcal{R})$ contains type 1 statements only, then there exists $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, $\mathcal{P}' - \mathcal{P}$ contains type 1 statements only, and for all $B.r \in \mathsf{Roles}(\mathcal{P})$, $[\![B.r]\!]_{\mathcal{P}'} = [\![B.r]\!]_{\mathcal{P}''}$.

Proof. There are 2 parts, each of which requires showing set inclusion in 2 directions.
Part 1 Under part (1), we let $\mathcal{P}'' = (ERR(\mathcal{P},\mathcal{R}) - (\mathcal{P} - \mathcal{P}')) \cup (\mathcal{P}' - \mathcal{P})$.
Part 1a We show that $[\![B.r]\!]_{\mathcal{P}'} \subseteq [\![B.r]\!]_{\mathcal{P}''}$. The proof uses induction on i to show that for all principals D, $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$ implies $D \in T_{\mathcal{P}''}\uparrow^\omega [\![B.r]\!]$. Consider the statement $B.r \leftarrow e$ used to introduce $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$. Observe that $\mathcal{P}' = (\mathcal{P}' - \mathcal{P}) \cup (\mathcal{P} - (\mathcal{P} - \mathcal{P}'))$. When the statement is in $\mathcal{P}' - \mathcal{P}$, it follows that it is in $\mathcal{P}''$ by construction, and so $D \in T_{\mathcal{P}''}\uparrow^\omega [\![B.r]\!]$ follows from the induction hypothesis. Consider the case in which the statement is in $\mathcal{P} - (\mathcal{P} - \mathcal{P}')$. Since $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$ and $T_{\mathcal{P}'}\uparrow^i$ is monotonic in i, each role $B'.r'$ in e is nonempty when evaluated under $\mathcal{P}'$. So Lemma 30 tells us that $[\![B.r]\!]_{UB(\mathcal{P},\mathcal{R})}$ is also nonempty. It follows from the definition of $ERR(\mathcal{P},\mathcal{R})$ that $B.r \leftarrow e$ is in $ERR(\mathcal{P},\mathcal{R})$, and the result again follows from the induction hypothesis.
Part 1b We show that $[\![B.r]\!]_{\mathcal{P}''} \subseteq [\![B.r]\!]_{\mathcal{P}'}$. The proof uses induction on i to show that for all principals D, $D \in T_{\mathcal{P}''}\uparrow^i [\![B.r]\!]$ implies $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$. Consider the statement $B.r \leftarrow e$ used to introduce $D \in T_{\mathcal{P}''}\uparrow^i [\![B.r]\!]$. By construction, such a statement must either be in $ERR(\mathcal{P},\mathcal{R}) - (\mathcal{P} - \mathcal{P}')$ or $\mathcal{P}' - \mathcal{P}$. When the statement is in $\mathcal{P}' - \mathcal{P}$, it follows that it is in $\mathcal{P}'$, and so $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$ follows from the induction hypothesis. When the statement is in $ERR(\mathcal{P},\mathcal{R}) - (\mathcal{P} - \mathcal{P}')$, then it must also be in $\mathcal{P} - (\mathcal{P} - \mathcal{P}')$, since $ERR(\mathcal{P},\mathcal{R}) \subseteq \mathcal{P}$. Observe that $\mathcal{P} - (\mathcal{P} - \mathcal{P}') = \mathcal{P} \cap \mathcal{P}'$. It follows that the $B.r \leftarrow e$ is in $\mathcal{P}'$, so it again follows that $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$ by using the induction hypothesis.
Part 2 Under part (2), we let $\mathcal{P}' = (\mathcal{P} - (ERR(\mathcal{P},\mathcal{R}) - \mathcal{P}'')) \cup (\mathcal{P}'' - ERR(\mathcal{P},\mathcal{R}))$.
Part 2a We show that $[\![B.r]\!]_{\mathcal{P}'} \subseteq [\![B.r]\!]_{\mathcal{P}''}$. The proof uses induction on i to show that for all principals D, $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$ implies $D \in T_{\mathcal{P}''}\uparrow^\omega [\![B.r]\!]$. Consider the statement $B.r \leftarrow e$ used to introduce $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$. By construction, such a statement must either be in $\mathcal{P} - (ERR(\mathcal{P},\mathcal{R}) - \mathcal{P}'')$ or $\mathcal{P}'' - ERR(\mathcal{P},\mathcal{R})$. When the statement is in $\mathcal{P}'' - ERR(\mathcal{P},\mathcal{R})$, it follows that it is in $\mathcal{P}''$, and so $D \in T_{\mathcal{P}''}\uparrow^\omega [\![B.r]\!]$ follows from the induction hypothesis. Now consider the case in which

$$B.r \leftarrow e \in \mathcal{P} - (ERR(\mathcal{P},\mathcal{R}) - \mathcal{P}'') \tag{5}$$

Next we show that $B.r \leftarrow e \in ERR(\mathcal{P},\mathcal{R})$. This holds because if $B.r \leftarrow e \notin ERR(\mathcal{P},\mathcal{R})$ then some $B'.r'$ appearing in e satisfies $T_{\mathcal{P}'}\uparrow^\omega [\![B'.r']\!] = \emptyset$, which violates our assumption that $B.r \leftarrow e$ is used to introduce $D \in T_{\mathcal{P}'}\uparrow^i [\![B.r]\!]$. Now since $ERR(\mathcal{P},\mathcal{R}) \subseteq \mathcal{P}$, we can rewrite (5) to $B.r \leftarrow e \in ERR(\mathcal{P},\mathcal{R}) - (ERR(\mathcal{P},\mathcal{R}) - \mathcal{P}'')$. This is is equivalent to $B.r \leftarrow e \in ERR(\mathcal{P},\mathcal{R}) \cap \mathcal{P}''$, which gives us $B.r \leftarrow e \in \mathcal{P}''$. From this $D \in T_{\mathcal{P}''}\uparrow^\omega [\![B.r]\!]$ now follows from the induction hypothesis.
Part 2b We show that $[\![B.r]\!]_{\mathcal{P}''} \subseteq [\![B.r]\!]_{\mathcal{P}'}$. The proof uses induction on i to show that for all principals D, $D \in T_{\mathcal{P}''}\uparrow^i [\![B.r]\!]$ implies $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$. Consider the statement $B.r \leftarrow e$ used to introduce $D \in T_{\mathcal{P}''}\uparrow^i [\![B.r]\!]$. Observe that $\mathcal{P}'' = (ERR(\mathcal{P},\mathcal{R}) - (\mathcal{P} - \mathcal{P}')) \cup (\mathcal{P}' - \mathcal{P})$. When the statement is in $\mathcal{P}' - \mathcal{P}$, it follows that it is in $\mathcal{P}'$, and so $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$ follows from the induction hypothesis. Consider the case in which the statement is in $ERR(\mathcal{P},\mathcal{R}) - (\mathcal{P} - \mathcal{P}')$. It must also be in $\mathcal{P} - (\mathcal{P} - \mathcal{P}')$, since $ERR(\mathcal{P},\mathcal{R}) \subseteq \mathcal{P}$. Observe that $\mathcal{P} - (\mathcal{P} - \mathcal{P}') = \mathcal{P} \cap \mathcal{P}'$. It follows that the $B.r \leftarrow e$ is in $\mathcal{P}'$, so it again follows that $D \in T_{\mathcal{P}'}\uparrow^\omega [\![B.r]\!]$ by using the induction hypothesis. ∎

Theorem 13 (Section 3.3) *Given an RCPI $\langle \mathcal{P},\mathcal{R},X.u \sqsupseteq A.r\rangle$, if $A.r \in \mathcal{G}_{\mathcal{R}} \cap \mathcal{S}_{\mathcal{R}}$, then $\mathcal{P}$ satisfies $X.u \sqsupseteq A.r$ under $\mathcal{R}$ if and only if $\mathcal{P}'$ satisfies $X.u \sqsupseteq \rho$ under $\mathcal{R}'$ for each $\langle \mathcal{P}',\mathcal{R}',X.u \sqsupseteq \rho\rangle \in$*

$Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$.

Proof. We prove entailment in both directions. We begin by showing the "only if" part, which we do by showing its contrapositive. That is, we assume there is a $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq \rho \rangle \in Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$ such that in some state $\mathcal{P}''$ reachable from $\mathcal{P}'$ under $\mathcal{R}'$, there is a principal E such that $E \in [\![\rho]\!]_{\mathcal{P}''}$, but $E \notin [\![X.u]\!]_{\mathcal{P}''}$. Using this assumption, we show $\mathcal{P}$ does not satisfy $X.u \sqsupseteq A.r$ under $\mathcal{R}$. There are now four cases given by the type of statement $A.r \leftarrow \alpha \in \mathcal{P}$ that is used to construct $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq \rho \rangle$.

Case 1: α is a principal. In this case $\rho = A'.r'$ and $[\![\rho]\!]_{\mathcal{P}''} = \{E\}$. Given the relationship between $\mathcal{R}$ and $\mathcal{R}'$, it is not hard to see that $\mathcal{P}''' = (\mathcal{P}'' - \{A'.r' \leftarrow E\})$ is reachable from $\mathcal{P}$ under $\mathcal{R}$. By the construction in $Decompose$ in this case, $A.r \leftarrow E \in \mathcal{P}$. Since $A.r \in \mathcal{S}_{\mathcal{R}}$, it follows that $A.r \leftarrow E \in \mathcal{P}'''$ and thus $E \in [\![A.r]\!]_{\mathcal{P}'''}$. Because $E \notin [\![X.u]\!]_{\mathcal{P}''}$, it follows by the monotonic nature of the semantics that $E \notin [\![X.u]\!]_{\mathcal{P}'''}$, completing the proof in this case.

Case 2: α is a role. In this case $\rho = \alpha$ and $E \in [\![\rho]\!]_{\mathcal{P}''}$ for some $\mathcal{P}''$ reachable from $\mathcal{P}$ under $\mathcal{R}$. By the construction in $Decompose$, in this case $E \in [\![A.r]\!]_{\mathcal{P}''}$ since $A.r \leftarrow \alpha \in \mathcal{P}''$ because, as above, $A.r \leftarrow \alpha \in \mathcal{P}$ and $A.r \in \mathcal{S}_{\mathcal{R}}$.

Case 3: α is a linked role of the form $B.r_1.r_2$. In this case $\rho = A'.r'$ and $E \in [\![\rho]\!]_{\mathcal{P}''}$. Given the relationship between $\mathcal{R}$ and $\mathcal{R}'$, it is not hard to see that $\mathcal{P}''' = (\mathcal{P}'' - \{A'.r' \leftarrow B.r_1.r_2\})$ is reachable from $\mathcal{P}$ under $\mathcal{R}$. By the construction in $Decompose$, $A'.r' \leftarrow \alpha$ is the only statement defining $A'.r'$ in $\mathcal{P}''$. Thus there exist some $C \in [\![B.r_1]\!]_{\mathcal{P}'''}$ such that $E \in [\![C.r_2]\!]_{\mathcal{P}'''}$. We also have $A.r \leftarrow B.r_1.r_2 \in \mathcal{P}'''$ because $A.r$ is shrink restricted so $E \in [\![A.r]\!]_{\mathcal{P}'''}$. Because $E \notin [\![X.u]\!]_{\mathcal{P}''}$, it follows by the monotonic nature of the semantics that $E \notin [\![X.u]\!]_{\mathcal{P}'''}$, completing the proof in this case.

Case 4: α is the intersection of two roles $B.r_1$ and $C.r_2$. In this case $\rho = A'.r'$ and $E \in [\![\rho]\!]_{\mathcal{P}''}$. Given the relationship between $\mathcal{R}$ and $\mathcal{R}'$, it is not hard to see that $\mathcal{P}''' = (\mathcal{P}'' - \{A'.r' \leftarrow B.r_1 \cap C.r_2\})$ is reachable from $\mathcal{P}$ under $\mathcal{R}$. By the construction in $Decompose$, $A'.r' \leftarrow \alpha$ is the only statement defining $A'.r'$ in $\mathcal{P}''$. Thus it must be the case that $E \in [\![B.r_1]\!]_{\mathcal{P}'''} \cap [\![C.r_2]\!]_{\mathcal{P}'''}$. We also have $A.r \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P}'''$ because $A.r$ is shrink restricted so $E \in [\![A.r]\!]_{\mathcal{P}'''}$. Because $E \notin [\![X.u]\!]_{\mathcal{P}''}$, it follows by the monotonic nature of the semantics that $E \notin [\![X.u]\!]_{\mathcal{P}'''}$, completing the proof in this the last case.

Now we show that entailment holds in the other direction, again by showing the contrapositive. We assume $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$ and there exists some principal E such that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$. Using this assumption, we show there exists at least one $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq \rho \rangle \in Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$ where $\mathcal{P}'$ does not satisfy $X.u \sqsupseteq \rho$ under $\mathcal{R}'$.

Because $E \in [\![A.r]\!]_{\mathcal{P}''}$ it must be the case that $E \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![A.r]\!]$, which means that (at least) one of the disjuncts in Definition 1 of $T_{\mathcal{P}''}(\pi)$ must hold when π is taken to be $T_{\mathcal{P}''}{\uparrow}^{\omega}$. Recall that $T_{\mathcal{P}''}{\uparrow}^{\omega} = T_{\mathcal{P}''}(T_{\mathcal{P}''}{\uparrow}^{\omega})$. Thus we have four cases, one for each disjunct.

Case 1: $E \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![A.r]\!]$ is generated by $A.r \leftarrow E \in \mathcal{P}''$. Since $A.r \in \mathcal{G}_{\mathcal{R}}$ it follows that $A.r \leftarrow E \in \mathcal{P}$. Thus there exists $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A'.r' \rangle \in Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$ in which $\mathcal{P}' = \mathcal{P} \cup \{A'.r' \leftarrow E\}$ and $\mathcal{R}' = \langle \mathcal{G}_{\mathcal{R}} \cup \{A'.r'\}, \mathcal{S}_{\mathcal{R}} \cup \{A'.r'\} \rangle$. Since $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$, it is easy to check that $\mathcal{P}''' = \mathcal{P}'' \cup A'.r' \leftarrow E$ is reachable from $\mathcal{P}'$ under $\mathcal{R}'$, and that $[\![A'.r']\!]_{\mathcal{P}'''} = \{E\}$. A simple induction on i shows that for all principals B_1 and B_2, and for all role names r_3, such that $B_1 \neq A'$ and $r_3 \neq r'$, $B_2 \in T_{\mathcal{P}''}{\uparrow}^{i}[\![B_1.r_3]\!]$ if and only if $B_2 \in T_{\mathcal{P}'''}{\uparrow}^{i}[\![B_1.r_3]\!]$. From this we get $E \notin [\![X.u]\!]_{\mathcal{P}'''}$ as required.

Case 2: $D \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![A.r]\!]$ is generated by $A.r \leftarrow B.r_1 \in \mathcal{P}'' \wedge E \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![B.r_1]\!]$. In this case we have $E \in [\![B.r_1]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$. By hypothesis $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$. Furthermore, by construction, $\langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq B.r_1 \rangle \in Decompose(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$, thus completing the proof in this case.

Case 3: $E \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![A.r]\!]$ is generated by $A.r \leftarrow B.r_1.r_2 \in \mathcal{P}'' \wedge \exists Z.\ Z \in T_{\mathcal{P}''}{\uparrow}^{\omega}[\![B.r_1]\!] \wedge E \in$

$T_{\mathcal{P}''}\uparrow^{\omega}[\![Z.r_2]\!]$. Fix such a Z. Since $A.r \in \mathcal{G}_{\mathcal{R}}$ it follows that $A.r \leftarrow B.r_1.r_2 \in \mathcal{P}$. Thus there exists $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A'.r' \rangle \in \mathit{Decompose}(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$ in which $\mathcal{P}' = \mathcal{P} \cup \{A'.r' \leftarrow B.r_1.r_2\}$ and $\mathcal{R}' = \langle \mathcal{G}_{\mathcal{R}} \cup \{A'.r'\}, \mathcal{S}_{\mathcal{R}} \cup \{A'.r'\} \rangle$. Since $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$, it is easy to check that $\mathcal{P}''' = \mathcal{P}'' \cup A'.r' \leftarrow E$ is reachable from $\mathcal{P}'$ under $\mathcal{R}'$, and that $E \in [\![A'.r']\!]_{\mathcal{P}'''}$. A simple induction on i shows that for all principals B_1 and B_2, and for all role names r_3, such that $B_1 \neq A'$ and $r_3 \neq r'$, $B_2 \in T_{\mathcal{P}''}\uparrow^i[\![B_1.r_3]\!]$ if and only if $B_2 \in T_{\mathcal{P}'''}\uparrow^i[\![B_1.r_3]\!]$. From this we get $E \notin [\![X.u]\!]_{\mathcal{P}'''}$, $m(B, r_1, Z) \in Z \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![B.r_1]\!]$, and $E \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![Z.r_2]\!]$. Since $A'.r' \leftarrow B.r_1.r_2 \in \mathcal{P}'''$, it follows that $E \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![A'.r']\!]$, giving us $E \in [\![A'.r']\!]_{\mathcal{P}'''}$ as required.

Case 4: $E \in T_{\mathcal{P}''}\uparrow^{\omega}[\![A.r]\!]$ is generated by $A.r \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P}'' \wedge E \in T_{\mathcal{P}''}\uparrow^{\omega}[\![B.r_1]\!] \wedge E \in T_{\mathcal{P}''}\uparrow^{\omega}$ $[\![C.r_2]\!]$. Since $A.r \in \mathcal{G}_{\mathcal{R}}$ it follows that $A.r \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P}$. Thus there exists $\langle \mathcal{P}', \mathcal{R}', X.u \sqsupseteq A'.r' \rangle \in \mathit{Decompose}(\mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r)$ in which $\mathcal{P}' = \mathcal{P} \cup \{A'.r' \leftarrow B.r_1.r_2\}$ and $\mathcal{R}' = \langle \mathcal{G}_{\mathcal{R}} \cup \{A'.r'\}, \mathcal{S}_{\mathcal{R}} \cup \{A'.r'\} \rangle$. Since $\mathcal{P}''$ is reachable from $\mathcal{P}$ under $\mathcal{R}$, it is easy to check that $\mathcal{P}''' = \mathcal{P}'' \cup A'.r' \leftarrow E$ is reachable from $\mathcal{P}'$ under $\mathcal{R}'$, and that $E \in [\![A'.r']\!]_{\mathcal{P}'''}$. A simple induction on i shows that for all principals B_1 and B_2, and for all role names r_3, such that $B_1 \neq A'$ and $r_3 \neq r'$, $B_2 \in T_{\mathcal{P}''}\uparrow^i[\![B_1.r_3]\!]$ if and only if $B_2 \in T_{\mathcal{P}'''}\uparrow^i[\![B_1.r_3]\!]$. From this we get $E \notin [\![X.u]\!]_{\mathcal{P}'''}$, $E \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![B.r_1]\!]$, and $E \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![C.r_2]\!]$. Since $A'.r' \leftarrow B.r_1 \cap C.r_2 \in \mathcal{P}'''$, it follows that $E \in T_{\mathcal{P}'''}\uparrow^{\omega}[\![A'.r']\!]$, giving us $E \in [\![A'.r']\!]_{\mathcal{P}'''}$ as required.

■

Lemma 16 (Equivalence of using $\mathcal{N}$ and $\mathcal{N}'$) (Section 4.1) Let $\pi = \langle \mathcal{P}, \mathcal{R}, X.u \sqsupseteq A.r \rangle$ be any RCPI. There exists $\mathcal{P}'$ and $E \in \mathsf{Principals}(\mathcal{P}')$ such that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}' \subseteq \mathcal{P} \cup \mathcal{N}(\pi)$, $E \in [\![A.r]\!]_{\mathcal{P}'}$, and $E \notin [\![X.u]\!]_{\mathcal{P}'}$ if and only if there exists $\mathcal{P}''$ and $E \in \mathsf{Principals}(\mathcal{P}'')$ such that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$, $E \in [\![A.r]\!]_{\mathcal{P}''}$, and $E \notin [\![X.u]\!]_{\mathcal{P}''}$.
Note that $\mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$ implies $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}''$.

Proof. The "if" part follows because, by construction, $\mathcal{N}'(\pi) \subseteq \mathcal{N}(\pi)$. For the "only if" part, suppose we are given any policy $\mathcal{P}'$ satisfying the conditions stated in the lemma. We show that $\mathcal{P}''$ exists satisfying the required conditions. We begin by constructing $\widehat{\mathcal{P}}$ from $\mathcal{P}'$ such that:

1. $E \in [\![A.r]\!]_{\widehat{\mathcal{P}}}$ and $E \notin [\![X.u]\!]_{\widehat{\mathcal{P}}}$;
2. $\widehat{\mathcal{P}}$ contains no statements of the form $F.r_1 \longleftarrow e$ such that $F.r_1 \in \mathsf{NoLinkedDefs}(\mathcal{P})$ $\lambda \longleftarrow F \notin \mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}}$;
3. $\widehat{\mathcal{P}}$ contains no statements of the form $\lambda \longleftarrow F$ such that $F \in \mathsf{NoLinkedPrinc}(\mathcal{P})$ and $\lambda \longleftarrow F \notin \mathcal{P}\restriction_{\mathcal{S}_{\mathcal{R}}}$.

The $\widehat{\mathcal{P}}$ we construct may not satisfy $\widehat{\mathcal{P}} \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$ because the construction may add principals not occurring in $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)$. In the second part of the proof, we will show how to obtain $\mathcal{P}''$ from $\widehat{\mathcal{P}}$ so that the size of $(\mathsf{Principals}(\mathcal{P}') - \mathsf{Principals}(\mathcal{P}))$ is less than or equal to the size of $\mathsf{NewPrinc}(\pi)$. The principals in the former set can then be replaced by principals in $\mathsf{NewPrinc}(\pi)$ according to any injective mapping to obtain the $\mathcal{P}'$ required.

As just mentioned the construction of $\widehat{\mathcal{P}}$ from $\mathcal{P}'$ introduces new principals not occurring in $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)$. In the construction below we use H to represent these new principals; it is important to understand that as we consider each $F \in \mathsf{Principals}(\mathcal{P})$, the new principal denoted by H is different for each different principal F.

The construction of $\widehat{\mathcal{P}}$ from $\mathcal{P}'$ proceeds as follows. For each $F \in \mathsf{Principals}(\mathcal{P})$ and each $r_1 \in \mathsf{Names}(\mathcal{P})$, if $F.r_1 \in \mathsf{NoLinkedDefs}(\mathcal{P})$, we select a new principal H not occurring in $\mathcal{R}$, $\mathcal{P}$, $\mathcal{P}'$, or in the partially constructed $\widehat{\mathcal{P}}$, and modify $\widehat{\mathcal{P}}$ by performing the following three steps. First, if

$F.r_1 \in \mathsf{NoLinkedDefs}(\mathcal{P})$ and $F.r_1 \longleftarrow e \in \mathcal{P}'$, add $H.r_1 \longleftarrow e$ to $\widehat{\mathcal{P}}$. If $F.r_1 \longleftarrow e \notin \mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}}$, additionally remove $F.r_1 \longleftarrow e$ from $\widehat{\mathcal{P}}$. This accomplishes requirement (1) above. Second, for each statement of the form $\lambda \longleftarrow F \in \mathcal{P}'$, add $\lambda \longleftarrow H$. If $F \in \mathsf{NoLinkedPrinc}(\mathcal{P})$ and $\lambda \longleftarrow F \notin \mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}}$, also remove $\lambda \longleftarrow F$. This accomplishes requirement (2) above. Third, for each statement $\lambda \longleftarrow e$ such that $F.r_1$ occurs in e, add $\lambda \longleftarrow e'$ in which e' is obtained from e by replacing $F.r_1$ by $H.r_1$. This step is required to ensure that each role other than the $F.r_1$ and $H.r_1$ considered above maintain the same role memberships in $\widehat{\mathcal{P}}$ as they do in $\mathcal{P}'$, except that if they contain F under $\mathcal{P}'$, they contain H under $\widehat{\mathcal{P}}$ and may or may not include F. We next clarify this point.

Intuitively, in $\widehat{\mathcal{P}}$, each new role $H.r_1$ and new principal H function exactly as the corresponding $F.r_1$ and F do in $\mathcal{P}'$. This can be made precise by proving that the role memberships obtained are identical for $\mathcal{P}'$ and $\widehat{\mathcal{P}}$, with the following three exceptions. (*i*) For the roles and principals $F.r_1$ and F considered above, every role that contains F in the semantics induced by $\mathcal{P}'$ contains H and may or may not contain F in the semantics induced by $\widehat{\mathcal{P}}$. (*ii*) Every role $F.r_1$ for which $H.r_1$ was introduced, $[\![H.r_1]\!]_{\widehat{\mathcal{P}}}$ is the same as $[\![F.r_1]\!]_{\mathcal{P}'}$, except that if the latter contains F, the former contains H and may or may not contain F, and (*iii*) $\widehat{\mathcal{P}}$ may or may not contains statements defining $F.r_1$. This can be shown by a straightforward induction on the fixpoint calculation. Note that because we have removed no statements in $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}}$ and have added only statements defining new principals not occurring in $\mathcal{G}_\mathcal{R}$, we have $\mathcal{P} \overset{*}{\underset{\mathcal{R}}{\mapsto}} \widehat{\mathcal{P}}$.

Having constructed $\widehat{\mathcal{P}}$, we now show how to eliminate redundant new principals so that the new principals in $\mathcal{P}''$ can be mapped injectively into principals in $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)$. Because this mapping is so straightforward, we suppress the details of it in the following.

Construction of $\mathcal{P}''$ such that $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$, $E \in [\![A.r]\!]_{\mathcal{P}''}$, and $E \notin [\![X.u]\!]_{\mathcal{P}''}$ follows by the same argument used by Li *et al.* in [21] in the proof of their Theorem 4.11. In that argument, an equivalence relation $\equiv$ over principals is introduced. Principals in $\mathcal{P}$ are equivalent only to themselves, while new principals are equivalent if they are members of the same subset of $\mathsf{SigRoles}(\pi)$. It is then shown that each member of the equivalence classes induced by $\equiv$ can be replaced by a single representative of the equivalence class. (It is assumed that the representative of the equivalence class containing E is E itself.) Note that this transformation does not remove any statement in $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}}$. The proof of Li *et al.* shows that from $E \in [\![A.r]\!]_{\widehat{\mathcal{P}}}$ and $E \notin [\![X.u]\!]_{\widehat{\mathcal{P}}}$ it follows that that $E \in [\![A.r]\!]_{\mathcal{P}''}$ and $E \notin [\![X.u]\!]_{\mathcal{P}''}$.

The important point for us here is that this construction of $\mathcal{P}''$ does not reintroduce any statements of the form removed from $\mathcal{P}'$ in the construction of $\widehat{\mathcal{P}}$. This holds because the statements removed during the construction of $\widehat{\mathcal{P}}$ define roles owned by principals in $\mathcal{P}$, while the representatives of the non-singleton equivalence classes do not occur in $\mathcal{P}$. Assuming the injective mapping of principals in $\mathcal{P}''$ to principals in $\mathsf{Principals}(\mathcal{P}) \cup \mathsf{NewPrinc}(\pi)$ has already been applied we now have $\mathcal{P}\restriction_{\mathcal{S}_\mathcal{R}} \subseteq \mathcal{P}'' \subseteq (\mathcal{P} \cup \mathcal{N}'(\pi))$, $E \in [\![A.r]\!]_{\mathcal{P}''}$, and $E \notin [\![X.u]\!]_{\mathcal{P}''}$, as required to complete the proof. ∎

Acknowledgements

William H. Winsborough is supported in part by NSF awards CNS-0716750, and CNS-0964710, and the Texas Higher Education Coordinating Board NHARP award 010115-0037-2007. Jianwei Niu is supported in part by NSF award CNS-0964710, the Texas Higher Education Coordinating Board NHARP award 010115-0037-2007, and University of Texas at San Antonio research award TRAC-2008.

References

[1] Cadence SMV. http://www.kenmcmil.com/smv.html.

[2] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat. Practical domain and type enforcement for unix. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 66–77, 1995.

[3] N. S. Barghouti, J. Mocenigo, and W. Lee. Grappa: A GRAPh PAckage in Java. In *GD '97: Proceedings of the 5th International Symposium on Graph Drawing*, pages 336–343, London, UK, 1997. Springer-Verlag.

[4] M. Y. Becker and P. Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *CSFW*, Washington, DC, USA, 2004. IEEE Computer Society.

[5] D. Bell and L. LaPadula. Computer security model: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, MITRE Corp., Bedford, MA, June 1975.

[6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[8] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[9] M. Drouineaud, M. Bortin, P. Torrini, and K. Sohr. A first step towards formal verification of security policy properties for RBAC. In *Proceedings of Fourth International Conference on Quality Software*, pages 60–67, 2004.

[10] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tshchantz. Verification and change-impact analysis of access-control policies. In *International Conference on Software Engineering*, pages 196–205. ACM Press, 2005.

[11] D. Gilliam, J. Powell, and M. Bishop. Application of lightweight formal methods to software security. In *14th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 160–165, 2005.

[12] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 167–176. IEEE Computer Society, 2005.

[13] F. Hansen and V. Oleshchuk. Conformance checking of RBAC policy and its implementation. In *Information Security Practice and Experience*, volume 3439 of *Lecture Notes in Computer Science*, pages 144–155. Springer Berlin/Heidelberg, 2005.

[14] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[15] T. Hobbs and W. Winsborough. Implementation and performance analysis of the role-based trust management system, RT^C. In M. Nishigaki, A. Jøsang, Y. Murayama, and S. Marsh, editors, *Trust Management IV*, volume 321 of *IFIP Advances in Information and Communication Technology*, pages 184–199. Springer Boston, 2010.

[16] S. Jha, N. Li, M. Tripunitara, Q. Wang, and W. Winsborough. Towards formal verification of role-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 5(4):242–255, 2008.

[17] S. Jha and T. Reps. Model checking SPKI/SDSI. *Journal of Computer Security*, 12:317–353, 2004.

[18] M. Koch, L. V. Mancini, and F. Parisi-Presicce. A graph-based formalism for RBAC. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):332–365, 2002.

[19] N. Li and J. C. Mitchell. Rt: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 01–212, April 2003.

[20] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.

[21] N. Li, J. C. Mitchell, and W. H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management. *Journal of the ACM*, 52(3):474–514, 2005.

[22] N. Li and M. V. Tripunitara. Security analysis in role-based access control. *ACM Transactions on Information and System Security*, 9(4):391–420, November 2006.

[23] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, 2003.

[24] R. J. Lipton and L. Snyder. A linear time algorithm for deciding subject security. *Journal of ACM*, 24(3):455–464, 1977.

[25] Z. Mao, N. Li, and W. H. Winsborough. Distributed credential chain discovery in trust management with parameterized roles and constraints (short paper). In *ICICS*, pages 159–173, 2006.

[26] M. J. May, C. A. Gunter, and I. Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*, pages 85–97, Washington, DC, USA, 2006. IEEE Computer Society.

[27] K. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic, 1993.

[28] M. Reith, J. Niu, and W. H. Winsborough. Role-based trust management security policy analysis and correction environment (RT-SPACE). In *ICSE Companion '08: Companion of the 30th International Conference on Software Engineering*, pages 929–930, 2008.

[29] M. Reith, J. Niu, and W. H. Winsborough. Toward practical analysis for trust management policy. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 310–321, 2009.

[30] R. L. Rivest and B. Lampson. Sdsi a simple distributed security infras- tructure, Oct. 1996. Available at http://theory.lcs.mit.edu/rivest/sdsi11.html.

[31] R. Sandhu. The typed access matrix model. In *Symposium on Research in Security and Privacy*, pages 122–136. IEEE Computer Society, 1992.

[32] R. S. Sandhu, E. J. Coyne, H. L. Feinstern, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[33] A. Sasturkar, P. Yang, S. D. Stoller, and C. R. Ramakrishnan. Policy analysis for administrative role based access control. In *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*, pages 124–138, Washington, DC, USA, 2006. IEEE Computer Society.

[34] A. Schaad, V. Lotz, and K. Sohr. A model-checking approach to analysing organisational controls in a loan origination process. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 139–149, New York, NY, USA, 2006. ACM Press.

[35] A. Schaad and J. Moffett. A lightweight approach to specification and analysis of role-based access control extensions. In *SACMAT '02: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 13–22, 2002.

[36] A. P. Sistla and M. Zhou. Analysis of dynamic policies. In *Proceedings of Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, pages 233–262, 2006.

[37] A. P. Sistla and M. Zhou. Analysis of dynamic policies. *Information and Computation*, 206(2-4):185–212, 2008.

[38] S. D. Stoller, P. Yang, M. Gofman, and C. R. Ramakrishnan. Symbolic reachability analysis for parameterized administrative role based access control. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 165–174, New York, NY, USA, 2009. ACM.

[39] S. D. Stoller, P. Yang, C. R. Ramakrishnan, and M. I. Gofman. Efficient policy analysis for administrative role based access control. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 445–455, New York, NY, USA, 2007. ACM.

[40] N. Zhang, M. Ryan, and D. P. Guelev. Synthesising verified access control systems in XACML. In *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 56–65. ACM Press, 2004.

[41] N. Zhang, M. Ryan, and D. P. Guelev. Evaluating access control policies through model checking. In *Proceedings of the 8th Information Security Conference*, volume 3650 of *Lecture Notes in Computer Science*, pages 446–460. Springer-Verlag, 2005.

[42] N. Zhang, M. Ryan, and D. P. Guelev. Synthesising verified access control systems through model checking. *Journal of Computer Security*, 16(1):1–61, 2008.