Beyond Proof-of-compliance: Safety and Availability Analysis in Trust Management

Ninghui Li
Department of Computer Science
Stanford University
Gates 4B
Stanford, CA 94305-9045
ninghui.li@cs.stanford.edu

William H. Winsborough
Network Associates Laboratories
15204 Omega Drive
Suite 300
Rockville, MD 20850-4601
william_winsborough@nai.com

John C. Mitchell
Department of Computer Science
Stanford University
Gates 4B
Stanford, CA 94305-9045
mitchell@cs.stanford.edu

Abstract

Trust management is a form of distributed access control using distributed policy statements. Since one organization may delegate partial control to another organization, it is natural ask what permissions may be granted as the result of policy changes by other organizations. We study security properties such as safety and availability for a family of trust management languages, devising algorithms for deciding the possible consequences of certain changes in policy. While trust management is more powerful in certain ways than mechanisms based on access control lists, and the security properties considered are more than simple safety, we find that in contrast to the classical HRU undecidability of safety properties, our primary security properties are decidable. In particular, most properties we studied are decidable in polynomial time. Containment, the most complicated security property we studied, is decidable in polynomial time for the simplest TM langauge in the family. The problem becomes intractable (co-NP-hard) when intersection or linked roles is added to the language.

1 Introduction

Trust Management is an approach to distributed access control and authorization, in which access control decisions are based on *policy statements* made by multiple principals. A TM language has a syntax for specifying policy statements and *queries* and a semantic relation $\vdash$. Given a set $\mathcal{P}$ of policy statements and a query[1] $\mathcal{Q}$, $\mathcal{P} \vdash \mathcal{Q}$ means that $\mathcal{Q}$ is true given $\mathcal{P}$. We call $\mathcal{P}$ a *state* of a TM system.

[1] In some systems, $\mathcal{Q}$ is required to be ground. Other systems allows $\mathcal{Q}$ to take the form $\exists \vec{x} \mathcal{Q}'(\vec{x})$, and one can find any (or every) $\vec{a}$ such that $\mathcal{P} \vdash \mathcal{Q}'(\vec{a})$.

The state of a TM system may change, new policy statements may be added and existing policy statements may be removed. In this paper, we study *security analysis* queries, which are questions about possible future states of a TM system. Often, one wants to restrict the possible ways to change the state of a TM system. Abstractly, we use $\mathcal{R}$ to denote a rule on what changes are allowed, and call it a *restriction rule*. Given $\mathcal{P}$ and $\mathcal{R}$, we write $\mathcal{P} \mapsto_{\mathcal{R}} \mathcal{P}'$ if $\mathcal{P}$ can change in one step to $\mathcal{P}'$ and the change is allowed by $\mathcal{R}$. We write $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ if $\mathcal{P}$ can change into $\mathcal{P}'$ in zero or more steps and every step is allowed by R, and say that the state $\mathcal{P}'$ is *R-reachable* from P (or simply *reachable*, when $\mathcal{P}$ and $\mathcal{R}$ are clear from context).

Definition 1 Assume we are given a set $\mathcal{P}$ of policy statements (the initial state), a restriction rule $\mathcal{R}$, and a query $\mathcal{Q}$. An *existential security analysis query* takes the following form: Does there exist $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and $\mathcal{P}' \vdash \mathcal{Q}$? When the above is true, we say $\mathcal{Q}$ is *possible* given $\mathcal{P}$ and $\mathcal{R}$. A *universal security analysis query* takes the following form: Is it true that for every $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, $\mathcal{P}' \vdash \mathcal{Q}$? When the above is true, we say $\mathcal{Q}$ is *necessary* given $\mathcal{P}$ and $\mathcal{R}$.

The following are some examples of security analysis queries.

Simple Safety (Existential) Whether there exists a reachable state in which a (presumably untrusted) principal has access to a resource.

Simple Availability (Universal) Whether in every reachable state, a (presumably trusted) principal has access to a resource.

Bounded Safety (Universal) Whether in every reachable state, the set of all principals that have access to a resource is bounded by a given set of principals.

Containment (Universal) Whether in every reachable state, every principal that has one property (e.g., has access to a resource) also has another property (e.g., being an employee). Such a query can be safety or availability (e.g., switching the two example properties in the previous sentence).

Simple safety queries were first formalized in [9] in the context of the well-known access matrix model [11]. The resulting model is commonly know as HRU. The problem of answering such queries is commonly known as *safety analysis*. In the general HRU model, safety analysis is undecidable [9]. A number of protection models were developed to address this, for example, the take-grant model [16], the Schematic Protection Model (SPM) [18], and the Typed Access Matrix model (TAM) [19]. The other kinds of queries listed above were not considered in these previous works. Since some queries are about availability rather than safety, we use the term *security analysis* rather than safety analysis.

Security analysis is especially relevant in TM systems, which decentralize control, for the following reasons. First, in TM systems, the set of policy statements one sees is often a subset of all the policy statements in the system. Therefore, if one wants to know the effect of its local policy statements, it is insufficient to just consider the policy statements one sees. One also needs to consider other possible policy statements that one does not know yet. In this case, one needs to use security analysis not because the state of the whole system may change in the future, but rather because one has only partial knowledge of the global state. Second, the key strength of TM languages is the support of delegation. Suppose that a resource owner delegates a right to another principal, who may issue arbitrary statements in the future. A natural concern is whether the resource owner still has some control over who can access the resources. Security analysis allows one to analyze how

much control one still maintains. To the best of our knowledge, security analysis for TM systems has not been investigated explicitly before.

In this paper, we define a precise model for security analysis in trust management. The policy language we consider is a slightly simplified (yet expressively equivalent) version of the RT_0 language [15]. We call the language SRT, for Simplified RT_0. All the security analysis queries listed above are considered. While the TM language we are studying supports delegation and is more expressive than the access matrix model in certain ways, and the kinds of queries we are considering are more general, somewhat surprisingly, answering these queries are decidable. Simple safety, simple availability, and bounded safety queries can be answered in time polynomial in the size of $\mathcal{P}$. Containment queries are the most expensive kind of queries. We show that, for BRT, a sub-language of SRT that has simple delegation, containment queries can be answered in polynomial time. Adding intersection to BRT increases the complexity of answering containment queries to co-NP-complete. For the case of SRT, which is obtained by adding an additional linked role feature, answering containment queries is still decidable (in co-NEXPtime).

The rest of this paper is organized as follows. In Section 2, we define the model we use to study security analysis in TM. In Section 3, we handle simple safety, simple availability, and bounded safety queries. In Section 4, we present results about containment queries. We compare with related work in Section 5, and conclude in Section 6. An appendix includes proofs not in the main body.

2 Instantiating The Security Analysis Problem

The abstract definition of security analysis in Definition 1 has three parameters: the language $\mathcal{P}$ is in, the form of query $\mathcal{Q}$, and the restriction rule $\mathcal{R}$. In this section, we instantiate the security analysis problem with specific parameters, discuss our choices, and give an example that will be used throughout the paper.

2.1 Syntax of The TM Langauge

The policy language we consider is SRT, a slightly simplified version of RT_0 [15]. As in RT_0, the basic constructs of SRT include *principals* and *role names*. In this paper, we use A, B, D, E, F, X, Y, and Z, sometimes with subscripts, to denote principals. A role name is an identifier, say, a string. We use r, u, and w, sometimes with subscripts, to denote role names. A *role* takes the form of a principal followed by a role name, separated by a dot, e.g., $A.r$ and $X.u$. A *role* defines a set of principals who are members of this role. Each principal A has the authority to define who are the members of each role of the form $A.r$. A role can also be viewed as an attribute. A principal is a member of a role if and only if it has the attribute identified by the role. An access control permission is represented as a role as well.

There are four types of policy statements in SRT, each corresponding to a different way of defining role membership:

- *Type-1*: $A.r \longleftarrow D$

  This statement means that A defines D to be a member of A's r role, or, in the attribute-based view, A says that D has the attribute r.

- *Type-2*: $A.r \longleftarrow B.r_1$

This statement means that A defines its r role to include (all members of) B's r_1 role. Or, if B says that a principal has the attribute r_1, then A says that it has the attribute r.

- *Type-3*: $A.r \longleftarrow A.r_1.r_2$

  We call $A.r_1.r_2$ a *linked role*. This statement means that A defines $A.r$ to include $B.r_2$ for every B that is a member of $A.r_1$. If r and r_2 are the same, A is delegating its authority over r to anyone that A believes to have the attribute r_1.

- *Type-4*: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$

  We call $B_1.r_1 \cap B_2.r_2$ an *intersection*. This statement means that A defines $A.r$ to include every principal who is a member of both $B_1.r_1$ and $B_2.r_2$.

A *role expression* is a principal, a role, a linked role, or an intersection. We say that each policy statement *defines* the role $A.r$. Given a set $\mathcal{P}$ of policy statements, we define the following: $\mathsf{Principals}(\mathcal{P})$ is the set of principals in $\mathcal{P}$, $\mathsf{Names}(\mathcal{P})$ is the set of role names in $\mathcal{P}$, and $\mathsf{Roles}(\mathcal{P})$ is the set of roles that can be constructed using principals in $\mathsf{Principals}(\mathcal{P})$ and role names in $\mathsf{Names}(\mathcal{P})$.

The language SRT simplifies RT_0 in that type-4 statements in SRT allow the intersection of only two roles; while in RT_0, the intersection may contain k components, each can be a principal, a role, or a linked role. This does not affect the expressive power, as statements using such intersection can be implemented by introducing new intermediate roles. This simplification helps simplify the proofs in this paper. When studying containment queries, we consider the following sub-languages of SRT: BRT (for Basic RT_0), which has only type-1 and 2 statements, LRT (for Linking RT_0), which has type-1, 2, and 3 statements, and NRT (for iNtersection RT_0), which has type-1, 2, 4 statements.

The four types of statements in SRT cover the most common delegation relationships in existing TM languages. SRT has the basic forms of delegation relationships in the RT framework [15, 14]. (SRT does not cover manifold roles in the RT framework.) The sub-language LRT can be viewed as a simplified version of SDSI [6, 4]. SDSI allows long linked names, which, as observed in [15], can be broken up by introducing new role names. The delegation relationships (not the s-expression-based representation of permission) in SPKI's 5-tuples, when not considering thresholds, can be captured by using type-1 statements and a restricted form of type-2 statements. That A delegates a permission r to B can be represented as $A.r \longleftarrow B$. That A allows B to further delegate this permission can be represented as $A.r \longleftarrow B.r$. As noted in [12], intersections can be used to implement threshold structures in SPKI. Similar analogies can be drawn for KeyNote [2].

Although SRT is limited in that role names can only be constants, extending role names in SRT to have internal structures does not change the nature of security analysis in a fundamental way. As we will see, security analysis is mostly affected by the structure of the delegation relationships. We believe that many results and techniques developed for SRT can be carried over to more expressive languages, e.g., RT_1 [14], which adds to RT_0 the ability to have parameterized roles, RT_1^C [13], which further adds constraints to RT_1, and, to a certain extent, SPKI/SDSI and KeyNote.

Third, the security analysis problem in SRT is quite nontrivial. Semantics and inferencing of SDSI, which is essentially the sub-language LRT, has been extensively studied [1, 4, 8, 10, 12, 15]. Some of these works only consider answering queries in a fixed state. Some considered universal queries where no restriction is placed on how the state may grow [1, 8]. The most interesting case of security analysis, i.e., answering queries with restrictions placed on state changes, is not studied in these previous works.

2.2 Semantics of the TM Langauge

We give a formal characterization of the meaning of a set $\mathcal{P}$ of policy statements by translating each policy statement in $\mathcal{P}$ into a datalog clause. (Datalog is a restricted form of logic programming (LP) with variables, predicates, and constants, but without function symbols.) We call the resulting program the semantic program of $\mathcal{P}$. We use the LP-based approach to define semantics because we will formulate safety computation rules using similar approach, and the LP-based approach generalizes easily to the case in which role names contain parameters, e.g., [14].

Definition 2 (Semantic Program) Given a set $\mathcal{P}$ of policy statements, the *semantic program*, $SP(\mathcal{P})$, of $\mathcal{P}$, has one ternary predicate m; $m(X, u, Z)$ represents that Z is a member of the role $X.u$. $SP(\mathcal{P})$ is constructed as follows. (Symbols that start with "?" represent logical variables.)

For each $A.r \longleftarrow D$ in $\mathcal{P}$	add	$m(A, r, D)$	$(m1)$
For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}$	add	$m(A, r, ?Z) :- m(B, r_1, ?Z)$	$(m2)$
For each $A.r \longleftarrow A.r_1.r_2$ in $\mathcal{P}$	add	$m(A, r, ?Z) :- m(A, r_1, ?Y),\ m(?Y, r_2, ?Z)$	$(m3)$
For each $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$ in $\mathcal{P}$	add	$m(A, r, ?Z) :- m(B_1, r_1, ?Z),\ m(B_2, r_2, ?Z)$	$(m4)$

A datalog program is a set of datalog clauses. Given a datalog program, $\mathcal{LP}$, its semantics can be defined through several equivalent approaches. The model-theoretic approach views $\mathcal{LP}$ as a set of first-order sentences and uses the minimal Herbrand model as the semantics. We write $SP(\mathcal{P}) \models m(X, u, Z)$ when $m(X, u, Z)$ is in the minimal Herbrand model of $SP(\mathcal{P})$. This semantics corresponds exactly to the set-theoretic semantics of RT_0 in [15].

We now summarize a standard fixpoint characterization of the minimal Herbrand model, which we will use in the proofs in this paper. For a datalog program, $\mathcal{LP}$, let $\mathcal{LP}^{inst}$ be the ground instantiation of $\mathcal{LP}$ using constants in $\mathcal{LP}$, the *immediate consequence operator*, $T_{\mathcal{LP}}$, is defined as follows. Given a set of ground logical atoms K, $T_{\mathcal{LP}}(K)$ consists of all logical atoms, a, such that $a :- b_1, \ldots, b_n \in \mathcal{LP}^{inst}$ and $b_j \in K$ for $1 \leq j \leq n$. The least fixpoint of $T_{\mathcal{LP}}$ can be constructed as follows. Define $T_{\mathcal{LP}}\uparrow^0 = \emptyset$ and $T_{\mathcal{LP}}\uparrow^{i+1} = T_{\mathcal{LP}}(T_{\mathcal{LP}}\uparrow^i)$ for $i \geq 0$. This defines an increasing sequence of subsets of a finite set. Thus there exists an N such that $T_{\mathcal{LP}}(T_{\mathcal{LP}}\uparrow^N) = T_{\mathcal{LP}}\uparrow^N$. $T_{\mathcal{LP}}\uparrow^N$ is easily shown to be the least fixpoint of $T_{\mathcal{LP}}$, which we denote by $T_{\mathcal{LP}}\uparrow^\omega$. $T_{\mathcal{LP}}\uparrow^\omega$ is identical to the minimal Herbrand model of $\mathcal{LP}$ [17]; therefore, $SP(\mathcal{P}) \models m(X, u, Z)$ if and only if $m(X, u, Z) \in T_{SP(\mathcal{P})}\uparrow^\omega$.

It has been shown that the minimal Herbrand model of $\mathcal{LP}$ can be computed in time linear in the size of $\mathcal{LP}^{inst}$ [5]. If the total size of $\mathcal{LP}$ is M, then there are $O(M)$ constants in $\mathcal{LP}$. Assuming that the number of variables in each clause is bounded by a constant, v, the number of instances of each clause is therefore $O(M^v)$, so the size of $\mathcal{LP}^{inst}$ is $O(M^{v+1})$. Therefore, the worst-case complexity of evaluating $SP(\mathcal{P})$ is $O(|\mathcal{P}|^3)$, since $|SP(\mathcal{P})| = O(|\mathcal{P}|)$ and each rule in $SP(\mathcal{P})$ has at most two variables.

2.3 Security Analysis Queries

In this paper, we consider the following three forms of security analysis queries.

- *Form-1*: $A.r \sqsupseteq \{D_1, \ldots, D_n\}$, where $n > 0$

  Intuitively, this means that all the principals $D_1, \ldots, D_n$ are members of $A.r$. Formally, $\mathcal{P} \vdash A.r \sqsupseteq \{D_1, \ldots, D_n\}$ if and only if $\{Z \mid SP(\mathcal{P}) \models m(A, r, Z)\} \supseteq \{D_1, \ldots, D_n\}$.

- *Form-2*: $\{D_1, \ldots, D_n\} \sqsupseteq A.r$

  Intuitively, this means that the member set of $A.r$ is bounded by the given set of principals. Formally, $\mathcal{P} \vdash A.r \sqsupseteq \{D_1, \ldots, D_n\}$ if and only if $\{D_1, \ldots, D_n\} \supseteq \{Z \mid SP(\mathcal{P}) \models m(A, r, Z)\}$.

- *Form-3* (role inclusion): $X.u \sqsupseteq A.r$

  Intuitively, this means that all the members of $A.r$ are also members of $X.u$. Formally, $\mathcal{P} \vdash X.u \sqsupseteq A.r$ if and only if $\{Z \mid SP(\mathcal{P}) \models m(X, u, Z)\} \supseteq \{Z \mid SP(\mathcal{P}) \models m(A, r, Z)\}$.

Simple safety and simple availability queries are special cases of form-1 queries; they have the form $A.r \sqsupseteq \{D\}$. The former is existential, and the latter is universal. Bounded safety queries are universal form-2 queries. Containment queries are universal form-3 queries.

2.4 Restriction Rules on State Changes

One important application of security analysis is in bounding the effect of policy-statement changes made by principals controlled by users within an organization. The trust management approach enables authority to be distributed among many principals, some of whom may be part of the same organization. The organization can use safety analysis in this context to require that principals within it only make changes that preserve desired properties, which are specified by a collection of safety queries. In this case, roles defined by principals within the organization can be viewed as unchanging, since the analysis will be repeated before any future candidate change is made to those roles. Roles defined by principals outside the organization, however, may change in arbitrary ways, since they are beyond the organization's control.

To model this control over roles, we use restriction rules of the form $\mathcal{R} = (\mathcal{G}_\mathcal{R}, \mathcal{S}_\mathcal{R})$, which consist of a pair of finite sets of roles. (In the rest of the paper we drop the subscripts from $\mathcal{G}$ and $\mathcal{S}$, as $\mathcal{R}$ is clear from context.)

- Roles in $\mathcal{G}$ are called *growth-restricted* (or *g-restricted*); no policy statements defining these roles can be added. Roles not in $\mathcal{G}$ are called *growth-unrestricted* (or *g-unrestricted*).

- Roles in $\mathcal{S}$ are called *shrink-restricted* (or *s-restricted*); policy statements defining these roles cannot be removed. Roles not in $\mathcal{S}$ are called *shrink-unrestricted* (or *s-unrestricted*).

One example of $\mathcal{R}$ is $(\emptyset, \emptyset)$, under which every role is g/s-unrestricted, i.e., both g-unrestricted and s-unrestricted. Under this R, all three forms of queries have trivial answers. Another example is $\mathcal{R} = (\emptyset, \mathsf{Roles}(\mathcal{P}))$, i.e., every role may grow unrestricted, and no statement defining roles in $\mathsf{Roles}(\mathcal{P})$ can be removed. Yet another example is $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, where $\mathcal{G} = \mathcal{S} = \{X.u \mid X \in \{X_1, \ldots, X_k\}, u \in \mathsf{Names}(\mathcal{P})\}$, i.e., $X_1, \ldots, X_k$ are trusted (controlled); therefore, every role $X.u$ such that $X \in \mathcal{T}$ is restricted, all other roles are unrestricted. If a principal X does not appear in $\mathcal{R}$, then for every role name r, by definition $X.r$ is *g/s-unrestricted*. This models the fact that the roles of unknown principals may be defined arbitrarily.

We allow some roles controlled by one principal to be g-restricted while other roles controlled by the same principal to be g-unrestricted. This provides more flexibility than simply identifying principals as trusted and untrusted, and permits one in practice to do security analysis only when changing certain roles. Similarly, we allow a role to be both g-restricted and s-unrestricted, which has the effect of making a safety check necessary when modifying the definition of the role only if adding a new statement.

The above kinds of restrictions are *static* in the sense that whether or not a state-change step is allowed by $\mathcal{R}$ does not depend on the current state. A dynamic restriction could, for instance, have $B.r_2$ be g-restricted if B is a member of $A.r_1$, which depends on the current state. Security analysis with dynamic restrictions is potentially interesting future work.

2.5 An Example

Example 1 The system admin of a company, SA, controls access to some resource, which we abstractly denote by SA.access. The company policy is the following: managers always have access to the resource, managers can delegate the access to other principals, but only to employees of the company. HR is trusted for issuing manager and employee credentials. The state $\mathcal{P}$ consists of the following statements:

SA.access ⟵ HR.employee ∩ SA.delegatedAccess
SA.delegatedAccess ⟵ SA.manager.access
HR.employee ⟵ HR.manager
HR.employee ⟵ Bob
Alice.access ⟵ Bob
SA.access ⟵ HR.manager
SA.manager ⟵ HR.manager
HR.manager ⟵ Alice
HR.employee ⟵ Carl

Given the above $\mathcal{P}$, Alice and Bob has access, Carl and Eve do not. One possible restriction rule has $\mathcal{S} =$ { SA.access, HR.manager } and $\mathcal{G} =$ { SA.access, HR.manager, HR.delegatedAccess, HR.emplyee }. We now list some example queries, together with the correct answers, give this rule:

A simply safety query: Is "SA.access ⊒ {Eve}" possible? (No.)
A simple availability query: Is "SA.access ⊒ {Alice}" necessary? (Yes.)
A bounded safety query: Is "{Alice, Bob} ⊒ SA.access" necessary. (No.)
A containment query: Is "HR.employee ⊒ SA.access" necessary? (Yes.)

3 Answering Form-1 and 2 Queries

SRT and its sub-languages are all monotonic in the sense that more statements will derive more role memberships (i.e., logical atoms of the form $m(A, r, D)$). This follows from the fact that the semantic program is a positive logic program. Form-1 queries are monotonic; given a form-1 query $\mathcal{Q}$, if $\mathcal{P} \vdash \mathcal{Q}$, then for every $\mathcal{P}'$ such that $\mathcal{P} \subseteq \mathcal{P}'$, $\mathcal{P}' \vdash \mathcal{Q}$. Form-2 queries are anti-monotonic; given a form-2 query $\mathcal{Q}$, if $\mathcal{P} \vdash \mathcal{Q}$, then for every $\mathcal{P}' \subseteq \mathcal{P}$, $\mathcal{P}' \vdash \mathcal{Q}$.

Intuitively, universal form-1 (simple availability) queries and existential form-2 queries can be answered by considering the lower bound of role memberships. A role's lower bound is the set of principals that are members of the role in every reachable state. Because $\mathcal{R}$ is static, there exists a minimal state that is reachable from $\mathcal{P}$ and $\mathcal{R}$, which is obtained from $\mathcal{P}$ by removing all statements defining s-unrestricted roles. We denote this state by $\mathcal{P}|_{\mathcal{R}}$. Clearly, $\mathcal{P}|_{\mathcal{R}}$ is reachable; furthermore, $\mathcal{P}|_{\mathcal{R}} \subseteq \mathcal{P}'$ for every reachable $\mathcal{P}'$. Since *SRT* is monotonic, one can compute the lower bound by computing the role memberships in $\mathcal{P}|_{\mathcal{R}}$.

Similarly, answering existential form-1 (simple safety) queries and universal form-2 (bounded safety) queries can be done by computing a "conceptual upper bound", which is the set of principals that can be members of the role in some reachable state. Intuitively, such bounds can be computed by considering a "maximal reachable state". However, such a "state" is not well-defined since it would contain an infinite set of policy statements, and we only allow a state to contain a finite set

of policy statements. We will show that one can simulate the "maximal reachable state" by a finite state and derive correct answers. We call role membership in this state the upper bound.

3.1 The Lower Bound

Definition 3 (The Lower Bound Program) Given $\mathcal{P}$ and $\mathcal{R}$, the *lower bound program* for them, $LB(\mathcal{P}, \mathcal{R})$, is constructed as follows:

For each $A.r \longleftarrow D$ in $\mathcal{P}\|_{\mathcal{R}}$	add	$lb(A, r, D)$	$(b1)$
For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}\|_{\mathcal{R}}$	add	$lb(A, r, ?Z) :- lb(B, r_1, ?Z)$	$(b2)$
For each $A.r \longleftarrow A.r_1.r_2$ in $\mathcal{P}\|_{\mathcal{R}}$	add	$lb(A, r, ?Z) :- lb(A, r_1, ?Y),\ lb(?Y, r_2, ?Z)$	$(b3)$
For each $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$ in $\mathcal{P}\|_{\mathcal{R}}$	add	$lb(A, r, ?Z) :- lb(B_1, r_1, ?Z),\ lb(B_2, r_2, ?Z)$	$(b4)$

The worst-case complexity of evaluating the lower bound program is $O(|\mathcal{P}|^3)$.

Observe that the above lower bound program is essentially the same as the semantic program for $\mathcal{P}|_{\mathcal{R}}$. They differ in that anywhere $LB(\mathcal{P}, \mathcal{R})$ uses the predicate lb, $SP(\mathcal{P}|_{\mathcal{R}})$ uses the predicate m. Therefore, we have the following fact.

Fact 3.1 *$LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D)$ if and only if $SP(\mathcal{P}|_{\mathcal{R}}) \models m(A, r, D)$.*

Proposition 3.2 *$LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D)$ if and only if for every $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, $SP(\mathcal{P}') \models m(A, r, D)$.*

Proof. The "only if" part (soundness) follows from the fact that $\mathcal{P}|_{\mathcal{R}} \subseteq \mathcal{P}'$ for every $\mathcal{P}'$ that is $\mathcal{R}$-reachable from $\mathcal{P}$, and that the language *SRT* is monotonic.

The "if" part (completeness) follows from Fact 3.1 and that $\mathcal{P}|_{\mathcal{R}}$ is reachable from $\mathcal{P}$. ∎

Proposition 3.2 means that the lower bound program can be used to answer universal form-1 queries and existential form-2 queries. We have not found an intuitive security meaning of existential form-2 queries, but include answering method for them here for completeness.

Corollary 3.3 *Given $\mathcal{P}$ and $\mathcal{R}$, a form-1 query $A.r \sqsupseteq \{D_1, \ldots, D_n\}$ is necessary if and only if $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D_i)$ for every i, $1 \leq i \leq n$.*

Corollary 3.4 *Given $\mathcal{P}$ and $\mathcal{R}$, a form-2 query $\{D_1, \ldots, D_n\} \sqsupseteq A.r$ is possible if and only if $\{D_1, \ldots, D_n\} \supseteq \{Z \mid LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, Z)\}$.*

Proof. For the "if" part, we must show that there exists $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and such that each D satisfying $\mathcal{P}' \models m(A, r, D)$ also satisfies $D \in \{D_1, \ldots, D_n\}$. It is easily seen by using Fact 3.1 that $\mathcal{P}|_{\mathcal{R}}$ is such a $\mathcal{P}'$. The "only if" part follows from Proposition 3.2 as follows. Suppose there exists Z such that $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, Z)$ and $Z \notin \{D_1, \ldots, D_n\}$. By Proposition 3.2, for every reachable $\mathcal{P}'$, $SP(\mathcal{P}') \models m(A, r, Z)$; therefore, the query is not possible. ∎

Consider Example 1. The simple availability query "is SA.access $\sqsupseteq$ {Alice} necessary" is true when both SA.access and HR.manager are all s-restricted, since then the two statements "SA.access ⟵ HR.manager" and "HR.manager ⟵ Alice" exist in the minimal state. On the other hand, it is not necessary that Bob has access, even when HR.employee is also s-restricted, since Alice could remove her statement "Alice.access ⟵ Bob".

3.2 The Upper Bound

To compute the upper bound of roles, we introduce the following notion: A role is *g-unbounded* if for each principal Z, there exists a reachable $\mathcal{P}'$ such that $\mathcal{P}' \vdash m(A, r, Z)$. In other words, $A.r$ could have every principal as its member. A g-unrestricted role is clearly g-unbounded. A g-restricted role may also be g-unbounded, as it may be defined to include a g-unbounded role.

The following fact about g-unbounded roles says that one only needs to consider one principal that does not occur in $\mathcal{P}$ (instead of every principal) to determine whether a role is g-unbounded.

Fact 3.5 *Given $\mathcal{P}$, $\mathcal{R}$, a role $A.r$, and a principal E that does not occur in $\mathcal{P}$, $A.r$ is g-unbounded if and only if there exists a reachable state $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A, r, E)$.*

See Appendix A.1 for the proof. We now show how to compute the upper bound, which simulate an infinite state.

Definition 4 (The Upper Bound Program) Given $\mathcal{P}$, $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, their upper bound program, $UB(\mathcal{P}, \mathcal{R})$, is constructed as follows. ($\top$ is a special principal symbol not occurring in $\mathcal{P}$, $\mathcal{R}$, or any query $\mathcal{Q}$.)

	add	$ub(\top, ?r, ?Z)$	(u)
For each $A.r \in \mathsf{Roles}(\mathcal{P}) - \mathcal{G}$	add	$ub(A, r, ?Z)$	$(u0)$
For each $A.r \longleftarrow D$ in $\mathcal{P}$	add	$ub(A, r, D)$	$(u1)$
For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}$	add	$ub(A, r, ?Z) :- ub(B, r_1, ?Z)$	$(u2)$
For each $A.r \longleftarrow A.r_1.r_2$ in $\mathcal{P}$	add	$ub(A, r, ?Z) :- ub(A, r_1, ?Y),\ ub(?Y, r_2, ?Z)$	$(u3)$
For each $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$ in $\mathcal{P}$	add	$ub(A, r, ?Z) :- ub(B_1, r_1, ?Z),\ ub(B_2, r_2, ?Z)$	$(u4)$

The computational complexity for evaluating $UB(\mathcal{P}, \mathcal{R})$ is $O(|\mathcal{P}|^3)$. Note that $\mathsf{Roles}(\mathcal{P})$ has $O(|\mathcal{P}|^2)$ elements, since there are $O(|\mathcal{P}|)$ principals and $O(|\mathcal{P}|)$ role names in $\mathcal{P}$. Therefore, there are $O(N^2)$ instance rules of $(u0)$; however, each such rule has only one variable.

Proposition 3.6 *Given any $\mathcal{P}$, $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, $A.r \in \mathsf{Roles}(\mathcal{P})$, and $Z \in \mathsf{Principals}(\mathcal{P}) \cup \{\top\}$, $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ if and only if there exists $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and $SP(\mathcal{P}') \models m(A, r, Z)$.*

See Appendix A.1 for the proof. From Fact 3.5 and Proposition 3.6, we have the following.

Corollary 3.7 *A role $X.u$ is g-unbounded if and only if $UB(\mathcal{P}, \mathcal{R}) \models ub(X, u, \top)$.*

Corollary 3.8 *Given $\mathcal{P}$ and $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, a form-1 query $A.r \supseteq \{D_1, \ldots, D_n\}$ is possible if and only if one of the following three conditions hold: (1) $A.r \notin \mathcal{G}$, (2) $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, \top)$, or (3) $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, D_i)$ for every i, $1 \leq i \leq n$.*

Proof. When $A.r \notin \mathsf{Roles}(\mathcal{P})$, the second and the third condition will not hold; the query is possible if and only if $A.r$ is g-unrestricted, i.e., $A.r \notin \mathcal{G}$. When $A.r \in \mathsf{Roles}(\mathcal{P})$, the first condition implies the second condition. Condition (2) or (3) both imply that the query is possible. If none of the three conditions holds, the query is not possible. Condition (2) is needed to deal with the case that some of the D_i's in the query do not occur in $\mathcal{P}$. ∎

Corollary 3.9 *Given $\mathcal{P}$ and $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, a form-2 query $\{D_1, \ldots, D_n\} \supseteq A.r$ is necessary if and only if $A.r \in \mathcal{G}$ and $\{D_1, \ldots, D_n\} \supseteq \{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\}$.*

Consider Example 1 again and assume we want the simple safety query "is SA.access $\sqsupseteq$ {Eve} possible" to be false. To achieve this without requiring Alice.access to be g-restricted, one has to make HR.employee g-restricted. In fact, simple safety is somewhat unnatural: to use it effectively, we have to be able to identify the principals that should never have access. Also, requiring that HR.employee be g-restricted is limiting, as this means that security analysis needs to be done every time a new employee is added. A more flexible approach would directly enforce that only employees get access; this can be done by using form-3 queries.

4 Answering Universal Form-3 Queries

Form-3 queries (i.e., role inclusion queries) are neither monotonic nor anti-monotonic. Given a form-3 query $X.u \sqsupseteq Z.w$ and three states $\mathcal{P}' \subseteq \mathcal{P} \subseteq \mathcal{P}''$, it is possible that $\mathcal{P} \vdash \mathcal{Q}$, but both $\mathcal{P}' \nvdash Q$ and $\mathcal{P}'' \nvdash Q$. As a result, the approach taken with form-1 and form-2 queries is not applicable here. We cannot simply look at a specific minimal (or maximal) state and answer the question.

In this paper, we restrict our attention to universal role inclusion queries. We have not found meaningful readings of existential role inclusion queries in terms of security properties. We say that a role $X.u$ *contains* another role $A.r$ if $X.u \sqsupseteq A.r$ is necessary, i.e., $X.u$ includes $A.r$ in every reachable state. And we call the problem of answering containment queries the *containment analysis* problem.

The case that one of $X.u$ and $A.r$ is not in $\mathsf{Roles}(\mathcal{P})$ is uninteresting. When $A.r \notin \mathsf{Roles}(\mathcal{P})$, then $X.u$ contains $A.r$ if and only if $A.r$ is g-restricted. If $X.u \notin \mathsf{Roles}(\mathcal{P})$ and $A.r \in \mathsf{Roles}(\mathcal{P})$, then $X.u$ contains $A.r$ if and only if $A.r$ has an upper bound that is empty. In the rest of this section, we only consider the case that both $X.u$ and $A.r$ are in $\mathsf{Roles}(\mathcal{P})$.

Intuitively, there are two cases in which a role $X.u$ contains a role $A.r$. The first case is that this containment is *forced* by the statements that are in $\mathcal{P}$. For example, if a statement $X.u \longleftarrow A.r$ exists and cannot be removed, then $X.u$ contains $A.r$. A containment may be forced by a chain of credentials. Forced containment can be computed similarly to role memberships.

In the second case, containment is caused by the nonexistence of statements in $\mathcal{P}$. In the extreme case, if $A.r$ has no definition and is g-restricted, then $A.r$ is contained in every role, since the member set of $A.r$ is empty in every reachable state. To compute this kind of containment, observe that a g-restricted $A.r$ is contained in another role $X.u$ if every definition of $A.r$ is contained in $X.u$. If $A.r$ has no definition at all, then it is contained in every role. However, a straightforward translation of this into a positive logic program does not work. Consider the following example: $\mathcal{P} = \{A.r \longleftarrow B.r_1,\ A.r \longleftarrow D,\ B.r_1 \longleftarrow A.r,\ X.u \longleftarrow D\}$ and $\mathcal{R} = (\{A.r, B.r_1\}, \{A.r, B.r_1, X.u\})$. In any $\mathcal{P}'$ that is $\mathcal{R}$-reachable from $\mathcal{P}$, the member sets of $A.r$ and $B.r_1$ are always $\{D\}$, and so both roles are contained by $X.u$. A straightforward positive logic program cannot derive this, since $X.u$ contains $A.r$ only if it contains $B.r_1$ and vice versa. As a result, neither containment relationship will be in the minimal model. To deal with this problem, we take the approach to prove non-containment using the minimal model of a logic program, and derive containment using negation-as-failure. Intuitively, $X.u$ contains $A.r$ unless we can find a witness entity E that is a member of $A.r$ in some state but not a member of $X.u$ in the same state.

Intuitively, containment queries that have the flavor of availability should be proven by forced containment. That a manager always has access to a resource should be due to a credential chain forcing this. In Example 1, this holds as long as SA.access is s-restricted. On the other hand, policy statements are unlikely to force everyone who has access to a resource to be an employee; the

orientation of the forced containment does not naturally correspond to this practical dependency. In Example 1, SA.employee contains SA.access as long as SA.access is g-restricted and HR.employee is s-unrestricted. This means that, as long as no new access rule is added and the statement "HR.employee $\longleftarrow$ HR.manager" is not removed, then the containment always holds.

4.1 Answering Containment Queries in *BRT*

Recall that the language *BRT* has only type-1 and type-2 statements.

Definition 5 (The Role Containment Program for *BRT*) Given a *BRT* state $\mathcal{P}$ and $\mathcal{R}$, the role containment program, $BCP(\mathcal{P}, \mathcal{R})$, includes the lower bound program $LB(\mathcal{P}, \mathcal{R})$ in Definition 3. In addition, it defines two predicates: $fc/4$ and $nc/4$. An atom $fc(X, u, Z, w)$ means that $X.u$ is forced to contain $Z.w$. An atom $nc(X, u, Z, w)$ means that $X.u$ does not contain $Z.w$. The program $BCP(\mathcal{P}, \mathcal{R})$ is derived from $LB(\mathcal{P}, \mathcal{R})$ as follows.

	add	$fc(?X, ?u, ?X, ?u)$	(c)
For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}\|_{\mathcal{R}}$	add	$fc(A, r, ?Z, ?w) :- fc(B, r_1, ?Z, ?w)$	$(c2)$
For each $A.r \in \mathsf{Roles}(\mathcal{P}) - \mathcal{G}$	add	$nc(?X, ?u, A, r) :- \sim fc(?X, ?u, A, r)$	$(n0)$
For each $A.r \in \mathcal{G}$, do the following:			
For each $A.r \longleftarrow D$ in $\mathcal{P}$	add	$nc(?X, ?u, A, r) :- \sim fc(?X, ?u, A, r),\ \sim lb(?X, ?u, D)$	$(n1)$
For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}$	add	$nc(?X, ?u, A, r) :- \sim fc(?X, ?u, A, r),\ nc(?X, ?u, B, r_1)$	$(n2)$

Rules (c) and $(c2)$ are straightforward. The intuition behinds $(n0)$ is that for $X.u$ to contain a g-unrestricted role $A.r$, $X.u$ has to be forced to contain $A.r$, since arbitrary new members may be added to $A.r$. The intuition behinds $(n1)$ is that, since $A.r$ contains D, if $X.u$'s lower bound does not contain D, and $X.u$ is not forced to contain $A.r$, then $X.u$ does not contain $A.r$. The "$\sim fc$" part is needed, since it may be the case that $A.r \longleftarrow D$ can be removed and $X.u \longleftarrow A.r$ exists and cannot be removed, in which case D may not be in $X.u$'s lower bound. Rule $(n2)$ means that $X.u$ does not contain $A.r$ if it does not contain $B.r_1$ and is not forced to contain $A.r$.

We now discuss the semantics of the logic program $BCP(\mathcal{P}, \mathcal{R})$, which uses negation-as-failure, but in a stratified manner. Given a logic program $\mathcal{LP}$, a predicate p (directly) depends on another predicate q if p is defined using q in the body. A predicate p negatively depends on q if $\sim q$ (the negation of q) is used to define p. For example, in $BCP(\mathcal{P}, \mathcal{R})$, fc depends on itself, nc depends on itself and negatively depends on fc and lb. A program is *stratified* if the predicates defined in it can be classified into strata such that each predicate depends only on predicates in the same or lower strata and negatively depends only on predicates in lower strata. A program without negation is trivially stratified, as no predicate depends negatively on any predicate at all. The program $BCP(\mathcal{P}, \mathcal{R})$ is also stratified. Predicates in the first stratum are lb and fc, and the only predicate in the second stratum is nc.

Most commonly accepted semantics for logic programming with negation-as-failure agree on stratified programs. Given a stratified datalog program $\mathcal{LP}$, let $\mathcal{LP}_1 \cup \cdots \cup \mathcal{LP}_n$ be a partition of $\mathcal{LP}^{Inst}$ such that $\mathcal{LP}_j$ consists of clauses defining predicates in the j'th stratum; we call $\mathcal{LP}_1 \cup \cdots \cup \mathcal{LP}_n$ a stratification of $\mathcal{LP}^{Inst}$. The semantics is obtained by first computing the minimal Herbrand model of $\mathcal{LP}_1$ and then use this model to determine truthfulness of negative literals in $\mathcal{LP}_2$ while computing a fixpoint for $\mathcal{LP}_1 \cup \mathcal{LP}_2$, and so on. Formally, we define an operator

Φ, which is parameterized by a ground logic program $\mathcal{LP}'$ and a set of ground atoms M. Given a set of ground logical atoms K, $\Phi_{\mathcal{LP}',M}(K)$ consists of all ground logic atoms, a, such that $a :- b_1, \ldots, b_n, \sim b_{n+1}, \ldots, \sim b_{n+m} \in \mathcal{LP}'$ and $b_i \in K$ and $b_{n+j} \notin M$. Given a logic program $\mathcal{LP}$ and $\mathcal{LP}_1 \cup \cdots \cup \mathcal{LP}_n$ a stratification of $\mathcal{LP}^{Inst}$, define $\Gamma^1_{\mathcal{LP}}$ to be $\Phi_{\mathcal{LP}_1,\emptyset}\uparrow^\omega$, i.e., the least fixpoint of $\Phi_{\mathcal{LP}_1,\emptyset}$. Define $\Gamma^{k+1}_{\mathcal{LP}}$ to be $\Phi_{\mathcal{LP}_1\cup\cdots\cup\mathcal{LP}_{k+1},\Gamma^k_{\mathcal{LP}}}\uparrow^\omega$ for $1 \leq k \leq n-1$. Then the model of $\mathcal{LP}$ is $\Gamma^n_{\mathcal{LP}}$. Each $\Gamma^i_{\mathcal{LP}}$ can be calculated in polynomial time, so the semantics of a stratified program can also be computed in polynomial time.

The following lemma says that the *fc* predicate in *BCP* is always sound for role containment, and it is complete when the second role is g-unrestricted.

Lemma 4.1 *Given a BRT state $\mathcal{P}$, $\mathcal{R}$, two roles $X.u$ and $A.r$, if $BCP(\mathcal{P},\mathcal{R}) \models fc(X,u,A,r)$, then $X.u$ contains $A.r$. If $X.u$ contains $A.r$ and $A.r$ is g-unrestricted, then $BCP(\mathcal{P},\mathcal{R}) \models fc(X,u,A,r)$.*

See Appendix A.2 for the proof. The following proposition says that role containment in *BRT* can be answered by using the program $BCP(\mathcal{P},\mathcal{R})$.

Proposition 4.2 *Given a BRT state $\mathcal{P}$, $\mathcal{R}$, and two roles $X.u$ and $A.r$ in* $\mathsf{Roles}(\mathcal{P})$*, $BCP(\mathcal{P},\mathcal{R}) \models nc(X,u,A,r)$ if and only if $X.u$ does not contain $A.r$.*

See Appendix A.2 for the proof.

4.2 Complexity Results for Containment Analysis in *NRT*, *LRT*, and *SRT*

NRT adds to *BRT* type-4 statements. Intersections in type-4 statements have the effect of conjunction. A role can be defined by multiple statements, which have the effect of disjunction. As a result, *NRT* can simulate formulas in propositional logic, and answering containment queries subsumes determining validity of propositional formulas, which is co-NP-complete.

Theorem 4.3 *Containment analysis in NRT is co-NP-complete.*

See Appendix A.3 for the proof. The co-NP-hard part is by reducing the monotone 3SAT problem, which is NP-complete, to the complement of containment analysis in *NRT*.

LRT adds to *BRT* type-3 statements. Linked roles in type-3 statements add the ability to simulate logical conjunction. Recall that the semantic rule for type-3 statements, $(m3)$, has a conjunction in the body, similar to that for type-4 statements, $(m4)$.

Theorem 4.4 *Containment analysis in LRT is co-NP-hard.*

See Appendix A.3 for the proof. We now give an upper bound on the computational complexity of containment analysis in *SRT*. This shows that containment analysis in *SRT* (and thus the sub-langauge *LRT*) is decidable.

Theorem 4.5 *Containment analysis in SRT is in co-NEXPtime.*

See Appendix A.3 for the proof.

5 Discussions and Related Work

We have shown that containment analysis is intractable in NRT, LRT, and SRT. This means that it is extremely unlikely that we will find an algorithm that is both sound and complete, and also has a worst-case polynomial time complexity. However, heuristic approaches are still possible. For example, it is straightforward to extend our LP-based approach for containment analysis in BRT to the case of LRT and SRT, such that containment relationships in Example 1 can be proved correctly. A possible approach is to use a sound but incomplete method and a complete but unsound method together to approximate the exact answer. Such a heuristic approach may be useful in practice, as it can give an exact answer in most cases. How to evaluate the effectiveness of such methods is interesting future work.

On the other hand, we have shown that in our TM model, simple safety queries can be solved efficiently. As discussed in Section 1, security analysis in the form of simple safety queries has been studied in the HRU model [9], and shown to be undecidable there. In this section we study the relationships between the two models, arguing informally that the HRU model does not include our TM model as a special case, and showing that there is an intuitive reason why security analysis in our model is decidable. We also seek to clarify the relationship between how trusted users are modelled in the two approaches. After this discussion of related work in safety analysis, we go on to discuss related work in trust management.

5.1 Comparison with the HRU Access Matrix Model

In the HRU model [9], a *protection system* has a finite set of rights and a finite set of commands. A *configuration* of a protection system is an access control matrix, with rows corresponding to subjects, and columns corresponding to objects; each cell in the matrix is a set of rights. A command takes the form of "if a list of conditions hold, execute a sequence of primitive operations." Each condition tests whether a right exists in a cell in the matrix. There are six kinds of primitive operations: enter a right into a specific cell in the matrix, delete a right from a cell in the matrix, create a new subject, create a new object, destroy an existing subject, and destroy an existing object. A command may be parameterized, with parameters being subjects or objects. In [9], Harrison et al. proved that for the HRU model, the safety question is undecidable, by showing that any Turing machine can be simulated by a protection system. For a fixed set of mono-operational commands, safety can be determined in time polynomial of the size of the access control matrix. However, if commands are a parameter to the problem, the safety problem is NP Complete.

In our model, given a state $\mathcal{P}$, the minimal Herbrand model of $SP(\mathcal{P})$ is a set of ground logical atoms. An atom $m(A, r, D)$ means that D is a member of A's r role. When A represents a resource, this can be viewed as D having the right r over A. Therefore, one can view principals as both subjects and objects and view role names as rights. This defines a straightforward mapping between the semantics of $\mathcal{P}$ and an access matrix. If all we have are type-1 statements, then adding (or removing) $A.r \longleftarrow D$ corresponds to adding (or removing) r to the cell on row D and column A. Adding a type-2 statement $A.r \longleftarrow B.r_1$ can be viewed as adding a trigger program, which for each row D, use parameters A, B, D to execute the following command: "$a2(X, Y, Z)$ { if r_1 in cell (Y, Z), add r to cell (X, Z) }". Note that this trigger program needs to be executed whenever the matrix changes. For example, if after $A.r \longleftarrow B.r_1$ is added, adding $B.r_1 \longleftarrow E$ will need to result in r being added to the cell (A, E). The statement $A.r \longleftarrow B.r_1$ gives B the power to add things

to A's column, which represents a delegation. Similarly, adding a type-3 statement $A.r \longleftarrow A.r_1.r_2$ can be viewed as adding a trigger program that executes the following command with parameters A, D, E for every D and E: "$a3(X, Y, Z)$ { if r_1 in cell (X, Y), and r_2 in cell (Y, Z), add r to cell (X, Z) }". Adding type-4 statement can be viewed in a similar manner. It is not clear how to model removing a statement using this approach.

There might be other ways of encoding our TM model in the HRU access matrix model, but the above encoding seems quite natural. From it, we make the following observations.

It seems unlikely that the HRU model subsumes the TM model as a special case. First, in the TM model, creating and removing principals are implicit. A principal can be viewed as created if it is used. A principal is considered removed if no statement mentions it. One could view the matrix as having an infinite number of rows and columns; however, only finitely many cells are nonempty. Second, one step of change in the TM model corresponds to executing many (one for every object when adding a type-2 or 4 statement, or one for every pair of objects in when adding a type-3 statement) simple commands in the HRU model. Third, triggers need to be used in order to achieve the effect of propagation. The last two are the main power of the TM model, and they do not exist in the HRU model.

That our TM model cannot subsume the HRU model is immediate from the complexity bounds. The underlying reason seems to be that the HRU commands we use to partially simulate our TM model have fixed schemas, instead of being arbitrary programs. As a result, we can exploit the properties of these fixed schemas. This seems to be the main reason that safety analysis, or the even more powerful containment analysis, is decidable in our model, but not in the HRU model.

Handling Trusted Subjects

Intuitively, a specific protection system is "safe" if access to resources without concurrence of the owner is impossible. However, protection systems often allow the owner to share rights to the resources. In that sense, they are not safe; the HRU model uses a weaker notion of safety: a user should be able to tell whether what he is about to do can lead to the further leakage of that right to untrusted subjects. The following is quoted from [9].

> To avoid a trivial "unsafe" answer because s himself can confer generic right r, we should in most circumstances delete s itself from the matrix. It might also make sense to delete from the matrix any other "reliable" subjects who could grant r, but whom s "trusts" will not do so. It is only by using the hypothetical safety test in this manner, with "reliable" subjects deleted, that the ability to test whether a right can be leaked has a useful meaning in terms of whether it is safe to grant a right to a subject.

Note that deleting a "reliable" subject from the matrix is stronger than stopping it from granting a right. Deleting a subject from the matrix will prevent the analysis from successfully simulating the execution of commands that check rights in the row or column corresponding the subject. However, it is inappropriate to ignore such commands: they may add undesirable rights and they may be initiated by "unreliable" subjects. In such cases, a system that is safe after the "reliable" subjects are removed is not safe in the actual system, even if "reliable" subjects do not initiate any command.

In our TM model, the restriction rule R represents the intuitive notion that certain principals are trusted. In practice, principals are controlled by users. When principals represent resources,

the controller is the subject who controls access to the resource. When principals represent public keys, the controller is the user who knows the private key.

5.2 Related Work in Trust Management

To our knowledge, no prior work investigates security analysis for trust management systems in the sense of verifying security properties that consider future state changes in which (parametric) restrictions are placed on allowed changes. In [3], a state transition model is used for comparing the expressive power of different access control mechanisms such as access control lists and trust management. There, security analysis is not the purpose. The language *SRT* is closely related to SDSI, whose semantics and evaluation has been the subject of many previous works [1, 4, 8, 10, 12, 15]. One main difference our work has is that we consider restricted state changes. We now list some similarities. The semantic approach we use is very similar to the semantics in [8]. Both [1] and [8] consider role inclusion queries in additional to membership queries. In some sense, they try to answer queries that hold when arbitrary new statements could be added, i.e., every role is g-unrestricted and s-restricted; the case that some roles are g-restricted is not considered. In [10], evaluating queries given a set of SDSI statements is reduced to model checking pushdown automata; there, only a fixed set of SDSI statements is considered, which are encoded as transition rules in the automata. Other works [4, 12, 15] do not handle role inclusion queries or consider restricted state changes.

6 Conclusion

Trust management systems such as *RT* allow independent principals to delegate partial authority over resources. While this is useful in many situations, delegation also raises the possibility of unanticipated and undesirable access. If Alice delegates access to her friend Bob, how can she be sure that Bob does not give permissions to her enemy Carol? We address this question by studying several forms of safety and availability properties, including general containment queries that capture both safety and availability.

Although the trust management primitives we consider are more expressive than some aspects of the HRU model [9], our main results show that persistence of nontrivial safety and availability properties may be algorithmically tractable. Specifically, form-1 queries and form-2 queries, both involving containment between a role and a fixed set of principals, can be answered using datalog programs that run in polynomial time. For general role inclusion queries, we look at several cases involving different policy sublanguages. For *BRT*, which only allows membership and delegation policy statements, containment for all reachable states is computable by a stratified datalog program with negation in polynomial time. For *NRT*, which is *BRT* plus intersection, the problem becomes co-NP-complete. Intuitively, the reason is that multiple statements about a role represent disjunction, while intersection of roles provides a corresponding form of conjunction. For *SRT*, which includes role linking, role containment for all reachable policy states remains decidable, but our current upper bound is co-NEXPtime (or double-exponential time).

We believe that security analysis is a critical problem for trust management. While combining policy statements from independent principals has practical appeal, the flexibility of distributed policy comes at a price. An individual or organization that owns a resource no longer has a direct way to determine who may be able to access the resource in the future. The key to providing

assurance to trust management users is to develop security analysis methods. The present paper identifies and solves certain security analysis problems, but much remains to be done. In more technical terms, the full impact of linked roles on containment analysis is not yet clear. Our complexity bounds on containment analysis for *LRT* and *SRT* are not tight. The proof that *LRT* is co-NP-hard uses roles that can shrink; we do not know whether the complexity changes if only role growth is allowed. Although containment analysis has no efficient algorithm in the worst case, there may be tractable subcases or useful heuristics. We also leave open for future work the consequences of more intricate restriction on policy changes. For example, it may be useful to impose restrictions that depend on the current policy, possibly formulated as policy invariants in some specification language.

A Proofs

A.1 Proofs of Fact 3.5 and Proposition 3.6

Fact 3.5: *Given $\mathcal{P}$, $\mathcal{R}$, a role $A.r$, and a principal E that does not occur in $\mathcal{P}$, $A.r$ is g-unbounded if and only if there exists a reachable state $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A, r, E)$.*

Proof. The "only if" part follows from the definition of g-unbounded roles.

In the "if" part, because *SRT* is monotonic, we can assume without loss of generality that $\mathcal{P}'$ is derived from $\mathcal{P}$ by adding some statements; let $\mathcal{P}' = \mathcal{P} \cup \mathcal{P}_1$. Given any principal Z, one can replace with Z all occurrence of E in the bodies of statements in $\mathcal{P}_1$, obtaining a new set of statements, $\mathcal{P}_2$. Let $\mathcal{P}'' = \mathcal{P}' \cup \mathcal{P}_2$. $\mathcal{P}''$ is reachable from $\mathcal{P}$ because it modifies the definitions of the same roles as does $\mathcal{P}'$. We show that $SP(\mathcal{P}'') \models m(A, r, Z)$ by using induction on i to show that for all $A.r$, if $m(A, r, E) \in T_{SP(\mathcal{P}')}\uparrow^i$, then $SP(\mathcal{P}'') \models m(A, r, Z)$. The basis is trivially satisfied because $T_{SP(\mathcal{P}')}\uparrow^0 = \emptyset$. In the step, $m(A, r, E) \in T_{SP(\mathcal{P}')}\uparrow^{i+1}$. This must be due to one of the four rules in $SP(\mathcal{P}')$, $(m1)$, $(m2)$, $(m3)$, or $(m4)$, which gives us the four following cases:

Case $(m1)$: $A.r \longleftarrow E \in \mathcal{P}'$. By construction of $\mathcal{P}''$, $A.r \longleftarrow Z \in \mathcal{P}''$. $SP(\mathcal{P}'') \models m(A, r, Z)$ follows from $(m1)$.

Case $(m2)$: $A.r \longleftarrow B.r_1 \in \mathcal{P}'$ and $m(B, r_1, E) \in T_{SP(\mathcal{P}')}\uparrow^i$. The induction hypothesis now gives us $SP(\mathcal{P}'') \models m(B, r_1, Z)$, from which $SP(\mathcal{P}'') \models m(A, r, Z)$ follows by $(m2)$.

Case $(m3)$: $A.r \longleftarrow A.r_1.r_2 \in \mathcal{P}'$ and $m(A, r_1, B), m(B, r_2, E) \in T_{SP(\mathcal{P}')}\uparrow^i$ for some B. The induction hypothesis now gives us $SP(\mathcal{P}'') \models m(B, r_2, Z)$. From $m(A, r_1, B) \in T_{SP(\mathcal{P}')}\uparrow^i$, we have $SP(\mathcal{P}') \models m(A, r_1, B)$, which gives us $SP(\mathcal{P}'') \models m(A, r_1, B)$ by monotonicity of *SRT*. We now have $SP(\mathcal{P}'') \models m(A, r, Z)$ by $(m3)$.

Case $(m4)$: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P}'$ and $m(B_1, r_1, E), m(B_2, r_2, E) \in T_{SP(\mathcal{P}')}\uparrow^i$. This case proceeds similarly to case $(m2)$ above. ∎

Proposition 3.6: *Given any $\mathcal{P}$, $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, $A.r \in \mathsf{Roles}(\mathcal{P})$, and $Z \in \mathsf{Principals}(\mathcal{P}) \cup \{\top\}$, $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ if and only if there exists $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and $SP(\mathcal{P}') \models m(A, r, Z)$.*

Proof. The "only if" part (Soundness): If $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$, consider $\mathcal{P}' = \mathcal{P} \cup \{X.u \longleftarrow Z \mid X.u \in \mathsf{Roles}(\mathcal{P}) - \mathcal{G}\}$. We show by induction on i that if $ub(A, r, Z) \in T_{UB(\mathcal{P},\mathcal{R})}\uparrow^i$, then $SP(\mathcal{P}') \models m(A, r, Z)$. The basis is trivial. In the step, $ub(A, r, Z) \in T_{UB(\mathcal{P},\mathcal{R})}\uparrow^{i+1}$, one of the rules in $UB(\mathcal{P}, \mathcal{R})$ is used to derive this. Case (u) is impossible, as $A \neq \top$. Case $(u0)$: $A.r \in \mathsf{Roles}(\mathcal{P}) - \mathcal{G}$,

by construction of $\mathcal{P}'$, $A.r \longleftarrow Z \in \mathcal{P}'$. So $SP(\mathcal{P}') \models m(A, r, Z)$ follows immediately by $(m1)$. Case $(u1)$: $A.r \longleftarrow Z \in \mathcal{P} \subseteq \mathcal{P}'$. In this case, $SP(\mathcal{P}') \models m(A, r, Z)$ also follows immediately by $(m1)$.

Case $(u2)$: $A.r \longleftarrow B.r_1 \in \mathcal{P} \subseteq \mathcal{P}'$ and $ub(B, r_1, Z) \in T_{UB(\mathcal{P},\mathcal{R})}\uparrow^i$. The induction assumption now gives us $SP(\mathcal{P}') \models m(B, r_1, Z)$, from which $SP(\mathcal{P}') \models m(A, r, Z)$ follows by $(m2)$.

Case $(u3)$: $A.r \longleftarrow A.r_1.r_2 \in \mathcal{P} \subseteq \mathcal{P}'$ and $ub(A, r_1, B), ub(B, r_2, Z) \in T_{UB(\mathcal{P},\mathcal{R})}\uparrow^i$ for some B. The induction assumption now gives us $SP(\mathcal{P}') \models m(A, r_1, B), m(B, r_2, Z)$, from which $SP(\mathcal{P}') \models m(A, r, Z)$ follows by $(m3)$.

Case $(u4)$: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P} \subseteq \mathcal{P}'$ and $ub(B_1, r_1, Z), ub(B_2, r_2, Z) \in T_{UB(\mathcal{P},\mathcal{R})}\uparrow^i$. The induction assumption now gives us $SP(\mathcal{P}') \models m(B_1, r_1, Z), m(B_2, r_2, Z)$, from which $SP(\mathcal{P}') \models m(A, r, Z)$ follows by $(m4)$.

The "if" part (Completeness): Suppose that there exists a reachable state $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A, r, Z)$. If $A.r \notin \mathcal{G}$, then $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ from $(u0)$. For the case in which $A.r \in \mathcal{G}$, we use induction on i to show that if $m(A, r, Z) \in T_{SP(\mathcal{P}')}\uparrow^i$, then $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$. The basis is trivial. In the step, there are four cases. Case $(m1)$: $A.r \longleftarrow Z \in \mathcal{P}'$. From $A.r \in \mathcal{G}$, we have $A.r \longleftarrow Z \in \mathcal{P}$. So $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ follows by using $(u1)$.

Case $(m2)$: $A.r \longleftarrow B.r_1 \in \mathcal{P}'$ and $m(B, r_1, Z) \in T_{SP(\mathcal{P}')}\uparrow^i$. The induction hypothesis gives us $UB(\mathcal{P}, \mathcal{R}) \models ub(B, r_1, Z)$, from which we obtain the desired $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ by $(u2)$.

Case $(m3)$: $A.r \longleftarrow A.r_1.r_2 \in \mathcal{P}'$ and $m(A, r_1, B), m(B, r_2, Z) \in T_{SP(\mathcal{P}')}\uparrow^i$ for some B. The induction hypothesis gives us $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r_1, B), ub(B, r_2, Z)$, from which we obtain the desired $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$ by $(u3)$.

Case $(m4)$: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P}'$ and $m(B_1, r_1, Z), m(B_2, r_2, Z) \in T_{SP(\mathcal{P}')}\uparrow^i$. This case is similar to the ones above. ∎

A.2 Proof of Lemma 4.1 and Proposition 4.2

We introduce the following terminology for the proof. The program $BCP(\mathcal{P}, \mathcal{R})$ has a stratification of two strata. Define BCP_1 to be the ground instantiation of clauses defining *lb* and *fc* in $BCP(\mathcal{P}, \mathcal{R})$, and BCP_2 to the ground instantiation of clauses defining *nc*. (We use BCP instead of $BCP(\mathcal{P}, \mathcal{R})$ for succinctness.) We write $BCP \models a$ if $a \in \Gamma^2_{BCP}$. When a is a ground instance of *fc* or *lb*, we write $BCP \models^i a$ if $a \in \Phi_{BCP_1,\emptyset}\uparrow^i$. When a is a ground instance of *nc*, we write $BCP \models^i a$ is $a \in \Phi_{BCP_1 \cup BCP_2, \Gamma^1_{BCP}}\uparrow^i$.

Lemma 4.1: *Given a BRT state $\mathcal{P}$, $\mathcal{R}$, two roles $X.u$ and $A.r$, if $BCP(\mathcal{P}, \mathcal{R}) \models fc(X, u, A, r)$, then $X.u$ contains $A.r$. If $X.u$ contains $A.r$ and $A.r$ is g-unrestricted, then $BCP(\mathcal{P}, \mathcal{R}) \models fc(X, u, A, r)$.*

Proof. Soundness: If $BCP \models fc(X, u, A, r)$, then there exists an integer i such that $BCP \models^i fc(X, u, A, r)$. Induction on i. The basis is trivial, as $\Phi_{BCP_1,\emptyset}\uparrow^0 = \emptyset$. Consider the step; either c or $(c2)$ is used to deduce that $BCP \models^{i+1} fc(X, u, A, r)$. Case (c): it must be that $X.u = A.r$, so it is trivial that $X.u$ contains $A.r$. Case $(c2)$: $X.u \longleftarrow B.r_1 \in \mathcal{P}|_{\mathcal{R}}$ and $BCP \models^i fc(B, r_1, A, r)$. By induction hypothesis, $B.r_1$ contains $A.r$. Furthermore, $X.u \longleftarrow B.r_1$ exists in every reachable state; therefore, $X.u$ contains $A.r$.

Completeness: Suppose $X.u$ contains $A.r$ and $A.r$ is g-unrestricted. Consider $\mathcal{P}' = \mathcal{P}|_{\mathcal{R}} \cup (A.r \longleftarrow E)$, in which E does not occur in $\mathcal{P}$. Observe that $X.u$ includes $A.r$ is true, since $\mathcal{P}'$ is reachable. Since $SP(\mathcal{P}') \models m(A, r, E)$, it must be that $m(X, u, E) \in T_{SP(\mathcal{P}')}\uparrow^i$ for some i. To complete the proof, we use induction on i to show that for each $Y.u$, if $m(Y, u, E) \in T_{SP(\mathcal{P})}\uparrow^i$, then $BCP \models fc(Y, u, A, r)$. Basis is trivial. In the step, one of $(m1)$ and $(m2)$ is

used to deduce that $m(Y,u,E) \in T_{SP(\mathcal{P}')}\uparrow^{i+1}$. Case $(m1)$: $Y.u \longleftarrow E \in \mathcal{P}'$, it must be that $Y.u = A.r$, since E does not occur in $\mathcal{P}$. From (c), $BCP \models fc(Y,u,A,r)$. Case $(m2)$: $Y.u \longleftarrow Y_1.u_1 \in \mathcal{P}'$, and $m(Y_1,u_1,E) \in T_{SP(\mathcal{P}')}\uparrow^i$. By definition of $\mathcal{P}'$, $Y.u \longleftarrow Y_1.u_1 \in \mathcal{P}|_{\mathcal{R}}$. From $(c2)$, $fc(Y,u,?Z,?w) :- fc(Y_1,u_1,?Z,?w) \in BCP$. By induction hypothesis, $BCP \models fc(Y_1,u_1,A,r)$, clearly $BCP \models fc(Y,u,A,r)$. ∎

Before proving Proposition 4.2, we first prove two auxiliary lemmas. Readers may wish to read the main proof first and refer to the two lemmas when they needed. The following lemma is used to prove the soundness of $(n1)$.

Lemma A.1 *Assume we are given $\mathcal{P}$ in BRT, $\mathcal{R}$, two roles $X.u$ and $A.r$, and a principal D such that $SP(\mathcal{P}|_{\mathcal{R}}) \not\models m(X,u,D)$. Let $\mathcal{P}' = \mathcal{P}|_{\mathcal{R}} \cup \{A.r \longleftarrow D\}$. If $SP(\mathcal{P}') \models m(X,u,D)$, then $BCP \models fc(X,u,A,r)$.*

Proof. We use induction on i to prove that for any $Z.w$ such that $SP(\mathcal{P}|_{\mathcal{R}}) \not\models m(Z,w,D)$, if $m(Z,w,D) \in T_{SP(\mathcal{P}')}\uparrow^i$, then $BCP \models fc(Z,w,A,r)$.

The basis is trivial. In the step, one of $(m1)$ and $(m2)$ is used to derive $m(Z,w,D) \in T_{SP(\mathcal{P}')}\uparrow^{i+1}$. Case $(m1)$: $Z.w \longleftarrow D \in \mathcal{P}'$. It must be that $Z.w = A.r$, since it cannot be that $Z.w \longleftarrow D \in \mathcal{P}|_{\mathcal{R}}$. By (c), $BCP \models fc(Z,w,A,r)$. Case $(m2)$: $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}'$ and $m(Z_1,w_1,D) \in T_{SP(\mathcal{P}')}\uparrow^i$. It follows that $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}|_{\mathcal{R}}$, by definition of $\mathcal{P}'$. And it follows that $SP(\mathcal{P}|_{\mathcal{R}}) \not\models m(Z_1,w_1,D)$, since otherwise $SP(\mathcal{P}|_{\mathcal{R}}) \models m(Z,w,D)$, which is contradictory. Now, by induction hypothesis, $BCP \models fc(Z_1,w_1,A,r)$, so the desired result holds by $(c2)$. ∎

The following lemma says that $(n2)$ is sound.

Lemma A.2 *Assume we are given a BRT state $\mathcal{P}$, $\mathcal{R}$, and three roles $X.u$, $A.r$, $B.r_1$, such that $A.r \longleftarrow B.r_1 \in \mathcal{P}$, $BCP(\mathcal{P},\mathcal{R}) \not\models fc(X,u,A,r)$, and $X.u$ does not contain $B.r_1$. Then $X.u$ does not contain $A.r$.*

Proof. Since $X.u$ does not contain $B.r_1$, there exists a reachable state $\mathcal{P}'$ and a principal E such that $SP(\mathcal{P}') \models m(B,r_1,E)$ and $SP(\mathcal{P}') \not\models m(X,u,E)$. We now construct a $\mathcal{P}''$ such that $SP(\mathcal{P}'') \models m(A,r,E)$ and $SP(\mathcal{P}'') \not\models m(X,u,E)$. $\mathcal{P}''$ is obtained from $\mathcal{P}'$ by first removing any $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}' - \mathcal{P}|_{\mathcal{R}}$ such that $SP(\mathcal{P}') \not\models m(Z_1,w_1,E)$, and then adding $A.r \longleftarrow B.r_1$. Clearly, $\mathcal{P}''$ is reachable. By induction on how $m(A,r,E)$ is proven in $SP(\mathcal{P}')$, it is easy to show that $SP(\mathcal{P}'') \models m(A,r,E)$.

To prove that $SP(\mathcal{P}'') \not\models m(X,u,E)$, we use induction on i to prove that for any $Z.w$ such that $SP(\mathcal{P}') \not\models m(Z,w,E)$, if $m(Z,w,E) \in T_{SP(\mathcal{P}'')}\uparrow^i$, then $BCP \models fc(Z,w,A,r)$. The basis is trivial. In the step, one of $(m1)$ and $(m2)$ is used to derive $m(Z,w,E) \in T_{SP(\mathcal{P}'')}\uparrow^{i+1}$. Case $(m1)$: $Z.w \longleftarrow E \in \mathcal{P}''$. This is impossible, as this means that $Z.w \longleftarrow E \in \mathcal{P}'$, which is contradictory with $SP(\mathcal{P}') \not\models m(Z,w,E)$. Case $(m2)$: $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}''$ and $m(Z_1,w_1,E) \in T_{SP(\mathcal{P}'')}\uparrow^i$. By definition of $\mathcal{P}''$, either $Z.w = A.r$ and $Z_1.w_1 = B.r_1$, or $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}'$. In the former case, $fc(Z,w,A,r)$ follows from (c). In the latter case, it follows that $SP(\mathcal{P}') \not\models m(Z_1,w_1,E)$, from $SP(\mathcal{P}') \not\models m(Z,w,E)$, and, by induction hypothesis, that $BCP \models fc(Z_1,w_1,A,r)$. Now the desired result holds by $(c2)$, provided we have $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}|_{\mathcal{R}}$. This follows from the construction of $\mathcal{P}''$ and the case assumption that $m(Z_1,w_1,E) \in T_{SP(\mathcal{P}'')}\uparrow^i$. ∎

Proposition 4.2: *Given a BRT state $\mathcal{P}$, $\mathcal{R}$, and two roles $X.u$ and $A.r$ in* Roles($\mathcal{P}$), $BCP(\mathcal{P},\mathcal{R}) \models nc(X,u,A,r)$ *if and only if $X.u$ does not contain $A.r$.*

Proof. The "only if" part (Soundness): We use induction on i to show that if $BCP \models^i nc(X,u,A,r)$, then $X.u$ does not contain $A.r$. Basis is trivial. In the step, one of $(n0)$, $(n1)$, and $(n2)$ is used to derive that $BCP \models^{i+1} nc(X,u,A,r)$. Case $(n0)$: $A.r$ must be g-unrestricted, and $BCP \models \sim fc(X,u,A,r)$; therefore, $BCP \not\models fc(X,u,A,r)$. From Lemma 4.1, $X.u$ does not contain $A.r$. Case $(n1)$: $A.r \longleftarrow D \in \mathcal{P}$, $BCP \models \sim lb(X,u,D)$, and $BCP \models \sim fc(X,u,A,r)$. Then $SP(\mathcal{P}|_{\mathcal{R}}) \not\models m(X,u,D)$ by Fact 3.1. Let $\mathcal{P}' = \mathcal{P}|_{\mathcal{R}} \cup \{A.r \longleftarrow D\}$. From Lemma A.1 it follows that $SP(\mathcal{P}') \not\models m(X,u,D)$; therefore $X.u$ does not contain $A.r$. Case $(n2)$: $A.r \longleftarrow B.r_1 \in \mathcal{P}$, $BCP \models^n nc(X,u,B,r_1)$, and $BCP \models \sim fc(X,u,A,r)$. By induction hypothesis, $X.u$ does not contain $B.r_1$; from Lemma A.2, $X.u$ does not contain $A.r$.

The "if" part (Completeness): If $X.u$ does not contain $A.r$, then we show that $BCP \models nc(X,u,A,r)$. When $A.r$ is g-unrestricted. From Lemma 4.1, $BCP \not\models fc(X,u,A,r)$, and so $BCP \models \sim fc(X,u,A,r)$. From $(n0)$, $BCP \models nc(X,u,A,r)$. In the rest of the proof, we only need to consider the case that $A.r$ is g-restricted. If $X.u$ does not contain $A.r$, then there exists a reachable state $\mathcal{P}'$ and a principal E such that $SP(\mathcal{P}') \models m(A,r,E)$ and $SP(\mathcal{P}') \not\models m(X,u,E)$. We use induction on i to show that if $m(A,r,E) \in T_{SP(\mathcal{P}')}\uparrow^i$, then $BCP \models nc(X,u,A,r)$. First observe that, from Lemma 4.1, it follows that $BCP \not\models fc(X,u,A,r)$, and so $BCP \models \sim fc(X,u,A,r)$. The basis is trivial. In the step, one of $(m1)$ and $(m2)$ is used to deduce that $m(A,r,E) \in T_{SP(\mathcal{P}')}\uparrow^{i+1}$. Case $(m1)$: $A.r \longleftarrow E \in \mathcal{P}'$, $A.r \longleftarrow E$ must be in $\mathcal{P}$ since $A.r$ is g-restricted. From Proposition 3.2 and $SP(\mathcal{P}') \not\models m(X,u,E)$, $BCP \not\models lb(X,u,E)$, and so $BCP \models \sim lb(X,u,E)$. From $(n1)$, $BCP(\mathcal{P},\mathcal{R}) \models nc(X,u,A,r)$. Case $(m2)$: $A.r \longleftarrow B.r_1 \in \mathcal{P}'$ and $m(B,r_1,E) \in T_{SP(\mathcal{P}')}\uparrow^i$. Since $A.r$ is g-restricted, $A.r \longleftarrow B.r_1 \in \mathcal{P}$. By induction hypothesis, $BCP(\mathcal{P},\mathcal{R}) \models nc(X,u,B,r_1)$. Therefore, $BCP(\mathcal{P},\mathcal{R}) \models nc(X,u,Z,w)$ by an instance of $(n2)$. ∎

A.3 Proofs of Theorems 4.3, 4.4, and 4.5

We first prove a lemma that will be used in establishing lower bounds on the complexity of containment analysis. The lemma says that if a containment does not hold, then there exists a counter-example state that only adds type-1 statements to $\mathcal{P}$ and only uses role names in $\mathcal{P}$.

Lemma A.3 *Given $\mathcal{P}$ and $\mathcal{R}$, two roles $X.u$ and $A.r$ in* Roles($\mathcal{P}$)*, if $X.u$ does not contain $A.r$, then there exists a $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A,r,E)$, $SP(\mathcal{P}') \not\models m(X,u,E)$, $\mathcal{P}' - \mathcal{P}$ only has type-1 statements, and $\mathcal{P}'$ only uses role names in $\mathcal{P}$.*

Proof. If $X.u$ does not contain $A.r$, then there exists a $\mathcal{P}'$ that $SP(\mathcal{P}') \models m(A,r,E)$ and $SP(\mathcal{P}') \not\models m(X,u,E)$. Given such a $\mathcal{P}'$, we first derive $\mathcal{P}''$ by replacing every statement $A.r \longleftarrow e \in \mathcal{P}' - \mathcal{P}$, where e is a role, a linked role, or an intersection, with a set of statements $\{A.r \longleftarrow Y \mid SP(\mathcal{P}') \models m(A,r,Y)\}$. Using induction, it is straightforward to show that the resulting state computes the exact same role memberships.

Now $\mathcal{P}'' - \mathcal{P}$ consists of only type-1 statements. From $\mathcal{P}''$, we derive $\mathcal{P}'''$ by removing all type-1 statements that uses role names (not roles) not appearing in $\mathcal{P}$. For example, a statement $A.v \longleftarrow D$ in $\mathcal{P}''$, where v does not appear in $\mathcal{P}$, will not be in $\mathcal{P}'''$. Using induction, it is straightforward to show that, for roles in Roles($\mathcal{P}$), $\mathcal{P}'''$ computes the exact same memberships as $\mathcal{P}''$. Intuitively,

$A.v \longleftarrow D$ can affect members of roles in $\mathsf{Roles}(\mathcal{P})$ unless some type-2, 3, or 4 statement refers to the role name v, which is impossible, since all type 2, 3, or 4 statements in $\mathcal{P}'$ are in $\mathcal{P}$, and so do not use v. ∎

Theorem 4.3: *Containment analysis in NRT is co-NP-complete.*

Proof. To show co-NP-hardness, we reduce the monotone 3SAT problem to the complement of the universal containment problem in *NRT*. Monotone 3SAT is 3SAT with each clause containing either only positive literals or only negative literals; it is known to be NP-complete [7].

Given an instance of monotone 3SAT: $\phi = c_1 \wedge \cdots \wedge c_\ell \wedge \overline{c_{\ell+1}} \wedge \cdots \wedge \overline{c_n}$, in which $c_1, \ldots, c_\ell$ are positive clauses and $\overline{c_{\ell+1}}, \ldots, \overline{c_n}$ are negative clauses. Let $p_1, \ldots, p_s$ be all the propositional variables in ϕ. For each negative clause $\overline{c_k} = (\neg p_{k_1} \vee \neg p_{k_2} \vee \neg p_{k_3})$, define $d_k = (p_{k_1} \wedge p_{k_2} \wedge p_{k_3})$, then $\overline{c_k} \Leftrightarrow \neg d_k$. Then $\phi \Leftrightarrow c_1 \wedge \cdots \wedge c_m \wedge \neg(d_{\ell+1} \vee \cdots \vee d_n)$. The formula ϕ is satisfiable if and only if $\psi = (c_1 \wedge \cdots \wedge c_\ell) \rightarrow (d_{\ell+1} \vee \cdots \vee d_n)$ is not valid. We now construct $\mathcal{P}$, $\mathcal{R}$, with the goal that $A.d \sqsupseteq A.c$ is necessary if and only if ψ is valid. In the construction, we use the role $A.p_i$ to denote the propositional variable p_i, $A.c_j$ to denote the clause c_j, and $A.d_k$ to denote the clause d_k. Define $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4$, in which

$\mathcal{P}_1 = \{A.c \longleftarrow A.c_1 \cap A.c'_1,\ A.c'_1 \longleftarrow A.c_2 \cap A.c'_2,\ \cdots,\ A.c'_{\ell-1} \longleftarrow A.c_{\ell-1} \cap A.c_\ell\}$.

$\mathcal{P}_2 = \{A.c_j \longleftarrow A.p_{j_1},\ A.c_j \longleftarrow A.p_{j_2},\ A.c_j \longleftarrow A.p_{j_3} \mid 1 \le j \le \ell,\ c_j = p_{j_1} \vee p_{j_2} \vee p_{j_3}\}$

$\mathcal{P}_3 = \{A.d \longleftarrow A.d_k \mid \ell + 1 \le k \le n\}$

$\mathcal{P}_4 = \{A.d_k \longleftarrow A.p_{k_1} \cap A.d'_k,\ A.d'_k \longleftarrow A.p_{k_2} \cap A.p_{k_3} \mid \ell + 1 \le k \le n,\ d_k = p_{k_1} \wedge p_{k_2} \wedge p_{k_3}\}$

Define R to be the restriction rule such that all the $A.p_i$'s are g-unrestricted and s-restricted, and all other roles are g/s-restricted.

We now show that $A.d \sqsupseteq A.c$ is *not* necessary if and only if ψ is *not* valid. First, the "only if" part: If $A.d \sqsupseteq A.c$ is not necessary, then there exists a reachable state $\mathcal{P}'$ and a principal E such that $SP(\mathcal{P}') \models m(A, c, E)$ and $SP(\mathcal{P}') \not\models m(A, d, E)$. Consider the truth assignment I defined as follows, for every i such that $1 \le i \le s$, $I(p_i) =$ true if $SP(\mathcal{P}') \models m(A, p_i, E)$, and $I(p_i) =$ false otherwise. Then under I, $(c_1 \wedge \cdots \wedge c_\ell)$ is true and $d_{\ell+1} \vee \cdots \vee d_n$ is false; therefore ψ is not valid. The "if" part: If ψ is not valid, then there exists a truth assignment I such that $(c_1 \wedge \cdots \wedge c_\ell)$ is true and $(d_{\ell+1} \vee \cdots \vee d_n)$ is false. Consider $\mathcal{P}' = \mathcal{P} \cup \{A.p_i \longleftarrow Z \mid 1 \le i \le s \wedge I(p_i) = \text{true}\}$. $\mathcal{P}'$ is reachable, and $SP(\mathcal{P}') \models m(A, c, Z)$ and $SP(\mathcal{P}') \not\models m(A, d, Z)$.

We now show that containment analysis in *NRT* is in co-NP. Given $\mathcal{P}$ and $\mathcal{R}$, if $X.u$ does not contain $A.r$, then there exists a reachable state $\mathcal{P}'$ and a principal E such that, $SP(\mathcal{P}') \models m(A, r, E)$ and $SP(\mathcal{P}') \not\models m(X, u, E)$. From Lemma A.3, we can assume, without loss of generality, that $\mathcal{P}' - \mathcal{P}$ consists of only type-1 statements and $\mathcal{P}'$ uses the same role names. From $\mathcal{P}'$, we construct $\mathcal{P}''$ as follows, let $\mathcal{P}'' = \mathcal{P}' \cap \mathcal{P} \cup \{Z.w \longleftarrow E \in \mathcal{P}' \mid Z.w \in \mathsf{Roles}(\mathcal{P})\}$. Clearly, $\mathcal{P}'' \subseteq \mathcal{P}'$ and $\mathcal{P}''$ is reachable. By induction on how $m(A, r, E)$ is proven in $SP(\mathcal{P}')$, it is easy to see that $SP(\mathcal{P}'') \models m(A, r, E)$. The size of $\mathcal{P}''$ is polynomial in $\mathcal{P}$. This means that if a containment does not hold, then there exists a short (polynomial in the size of the input program $\mathcal{P}$) counterproof such that one can check in polynomial time. This shows that the problem is in co-NP. The method we use to construct the counter example $\mathcal{P}''$ also yields an exponential algorithm for determining containment. ∎

Theorem 4.4 *Containment analysis in LRT is co-NP-hard.*

Proof. As in the proof of Theorem 4.3, we reduce the monotone 3SAT problem to the complement of universal role containment in *LRT*. Similarly, given an instance ϕ of monotone 3SAT, we

construct $\psi = (c_1 \wedge \cdots \wedge c_\ell) \rightarrow (d_{\ell+1} \vee \cdots \vee d_n)$ such that ϕ is satisfiable if and only if ψ is not valid.

We now construct $\mathcal{P}$, $\mathcal{R}$, such that $A.d \sqsupseteq A.c$ is necessary if and only if ψ is valid. Define $\mathcal{P}$ to be $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4 \cup \mathcal{P}_5$, in which

$\mathcal{P}_1 = \{A.c \longleftarrow A.c'_1.c_1,\ A.c'_1 \longleftarrow A.c'_2.c_2,\ \cdots,\ A.c'_{\ell-2} \longleftarrow A.c'_{\ell-1}.c_{\ell-1},\ A.c'_{\ell-1} \longleftarrow A.c_\ell\}$
$\mathcal{P}_2 = \{A.c_j \longleftarrow A.p_{j_1},\ A.c_j \longleftarrow A.p_{j_2},\ A.c_j \longleftarrow A.p_{j_3} \mid 1 \leq j \leq \ell,\ c_j = p_{j_1} \vee p_{j_2} \vee p_{j_3}\}$
$\mathcal{P}_3 = \{A.d \longleftarrow A.d_k \mid \ell + 1 \leq k \leq n\}$
$\mathcal{P}_4 = \{A.d_k \longleftarrow A.d'_k.p_{k_1},\ A.d'_k \longleftarrow A.p_{k_2}.p_{k_3} \mid \ell + 1 \leq k \leq n,\ d_k = p_{k_1} \wedge p_{k_2} \wedge p_{k_3}\}$
$\mathcal{P}_5 = \{A.p_i \longleftarrow A \mid 1 \leq i \leq s\}$

Let R be the restriction rule such that all the $A.p_i$'s are g-restricted and s-unrestricted, and all other roles mentioned in $\mathcal{P}$ are g/s-restricted.

In every reachable state, the definitions of some $A.p_i$'s are removed, which correspond to assigning false to some of the p_i's. In every reachable state, $A.c$ and $A.d$ either includes only A or is empty. $A.c$ includes A if and only if the corresponding truth assignment makes $c_1 \wedge \cdots \wedge c_m$ true, and $A.d$ includes A if and only the corresponding truth assignment makes $(d_{m+1} \vee \cdots \vee d_n)$ true. Therefore, $A.c$ contains $A.d$ if and only if ψ is valid. ∎

Theorem 4.5: *Containment analysis in SRT is in co-NEXPtime.*

Proof. Given $\mathcal{P}$ and $\mathcal{R}$, if a query $X.u \sqsupseteq A.r$ is not necessary, i.e., $X.u$ does not contain $A.r$, then there exists a reachable state $\mathcal{P}'$ and a principal E such that $SP(\mathcal{P}') \models m(A, r, E)$ and $SP(\mathcal{P}') \not\models m(X, u, E)$. From Lemma A.3, we can assume, without lose of generality, that $\mathcal{P}' - \mathcal{P}$ consists of only type-1 statements and $\mathcal{P}'$ uses the same role names as $\mathcal{P}$.

Given such a $\mathcal{P}'$ and E, we show that one can construct another state $\mathcal{P}''$ that has size exponential in $\mathcal{P}$ and $SP(\mathcal{P}'') \models m(A, r, E)$ and $SP(\mathcal{P}') \not\models m(X, u, E)$. The way we construct $\mathcal{P}''$ is through collapsing equivalent principals in $\mathcal{P}'$ into one, to be made precise as follows. Let $\mathsf{SigRoles}(\mathcal{P}, \mathcal{P}', \mathcal{Q})$ be $\{X.u\} \cup \{A.r_1 \mid A.r \longleftarrow A.r_1.r_2 \in \mathcal{P} \cap \mathcal{P}'\} \cup \{B_1.r_1, B_2.r_2 \mid A.r \longleftarrow B_1.r_1 \cap B_2.r_2 \in \mathcal{P} \cap \mathcal{P}'\}$. Define a binary relation $\equiv$ over the principals in $\mathcal{P}'$ as follows: $Y_1 \equiv Y_2$ if one of the following two conditions are satisfied: (1) $Y_1 = Y_2$; (2) $Y_1, Y_2 \notin \mathsf{Principals}(\mathcal{P})$ and for every role $Z.w \in \mathsf{SigRoles}(\mathcal{P}, \mathcal{P}', \mathcal{Q})$, $SP(\mathcal{P}') \models m(Z, w, Y_1)$ if and only if $SP(\mathcal{P}') \models m(Z, w, Y_2)$. The relation $\equiv$ is easily seen to be an equivalence relation. For each equivalence class, we pick one principal in it as a unique representative; for a given principal Y, we use $[Y]$ to denote the representative the equivalence class of Y. We assume that $[E] = E$. $\mathcal{P}''$ is constructed from $\mathcal{P}'$ as follows: for each statement, replace all the principals with their representatives, then remove duplicate statements.

Given $\mathcal{P}$ that has size N, clearly $\mathsf{SigRoles}(\mathcal{P}, \mathcal{P}', \mathcal{Q})$ has $O(N)$ roles. Therefore, there are in total $M = O(2^{O(N)})$ principals in $\mathcal{P}'$, these principals will result in $O(M^2 N)$ new type-1 statements. Therefore, if a containment does not hold, there exists a counter-example state that has size exponential in $\mathcal{P}$. Once the state is guessed correctly, it can be verified in time polynomial in the size of the state. This shows that the problem is in co-NEXPtime. An obvious algorithm that has double exponential time complexity is as follows: first collect $\mathsf{SigRoles}(\mathcal{P}, \mathcal{P}, \mathcal{Q})$ from $X.u$ and all type-2 and 3 statements from $\mathcal{P}$, and add one principal for each subset of $\mathsf{SigRoles}(\mathcal{P}, \mathcal{P}, \mathcal{Q})$, then enumerate all reachable sub-states to see whether a containment holds.

It remains to prove that our construction of $\mathcal{P}''$ works, i.e., that $SP(\mathcal{P}'') \models m(A, r, E)$ and $SP(\mathcal{P}'') \not\models m(X, u, E)$.

To prove $SP(\mathcal{P}'') \models m(A, r, E)$, we use induction to prove the following claim: For any role $Z.w$ in $\mathsf{Roles}(\mathcal{P}')$ and Y in $\mathcal{P}'$, if $m(Z, w, Y) \in T_{SP(\mathcal{P}')} \uparrow^i$, then $SP(\mathcal{P}'') \models m([Z], w, [Y])$. The

basis is trivial, since $T_{SP(\mathcal{P}')}\uparrow^0 = \emptyset$. Now consider the step. One of $(m1)$, $(m2)$, $(m3)$, and $(m4)$ is used to derive $m(Z, w, Y) \in T_{SP(\mathcal{P}')}\uparrow^{i+1}$. Case $(m1)$: $Z.w \longleftarrow Y \in \mathcal{P}'$. By construction of $\mathcal{P}''$, $[Z].w \longleftarrow [Y] \in \mathcal{P}''$; therefore, $SP(\mathcal{P}'') \models m([Z], w, [Y])$. In the next three cases, a type-2, 3, or 4 statement $A.r \longleftarrow e$ exist in $\mathcal{P}'$. It must also exist in $\mathcal{P}$, since $\mathcal{P}' - \mathcal{P}$ only has type-1 statements; therefore, principals in $A.r \longleftarrow e$ are each in their own equivalence class. The statement must also exist in $\mathcal{P}''$, since the equivalence substitution for $A.r \longleftarrow e$ will not change the statement. Case $(m2)$: $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}', \mathcal{P}, \mathcal{P}''$ and $m(Z_1, w_1, Y) \in T_{SP(\mathcal{P}')}\uparrow^i$. From induction hypothesis, $SP(\mathcal{P}'') \models m([Z_1], w_1, [Y])$. It must be that $[Z_1] = Z_1$. The claim then follows from $(m2)$. Case $(m3)$: $Z.w \longleftarrow Z.w_1.w_2 \in \mathcal{P}', \mathcal{P}, \mathcal{P}''$ and $m(Z, w_1, F), m(F, w_2, Y) \in T_{SP(\mathcal{P}')}\uparrow^i$. It must be that $[Z] = Z$. By induction hypothesis, $SP(\mathcal{P}'') \models m([Z], w_1, [F])$, and $SP(\mathcal{P}'') \models m([F], w_2, [Y])$. The claim follows from $(m3)$. Case $(m4)$: $Z.w \longleftarrow Z_1.w_1 \cap Z_2.w_2 \in \mathcal{P}', \mathcal{P}, \mathcal{P}''$ and $m(Z_1, w_1, Y), m(Z_2, w_2, Y) \in T_{SP(\mathcal{P}')}\uparrow^i$. This case is similar to $(m2)$.

We now prove that $SP(\mathcal{P}'') \not\models m(X, u, E)$, by proving the following claim: for any role $Z.w \in \mathsf{Roles}(\mathcal{P}'')$ and any principal Y in $\mathsf{Principals}(\mathcal{P}'')$, if $m(Z, w, Y) \in T_{SP(\mathcal{P}'')}\uparrow^i$, then there exists Z', Y' such that $[Z'] = Z$ and $[Y'] = Y$ and $SP(\mathcal{P}') \models m(Z', w, Y')$. Given this claim, if $SP(\mathcal{P}'') \models m(X, u, E)$, then there exists X' and E' in $\mathsf{Principals}(\mathcal{P}')$ such that $[X'] = X$, $[E'] = E$, and $SP(\mathcal{P}') \models m(X', u, E')$. Since $X \in \mathsf{Principals}(\mathcal{P})$, it must be that $X' = X$. And by definition of $\equiv$, $[E'] = E$ means that E is also a member of $X.u$, giving us a contradiction with our assumption on $\mathcal{P}'$.

We now use induction to prove the claim. The basis is trivial, since $T_{SP(\mathcal{P}'')}\uparrow^0 = \emptyset$. Now consider the step. One of $(m1)$, $(m2)$, $(m3)$, and $(m4)$ is used to derive $m(Z, w, Y) \in T_{SP(\mathcal{P}'')}\uparrow^{i+1}$. Case $(m1)$: $Z.w \longleftarrow Y \in \mathcal{P}''$. By definition of $\mathcal{P}''$, there exists $Z'.w \longleftarrow Y' \in \mathcal{P}'$ such that $[Z'] = Z$ and $[Y'] = [Y]$. From this we have $SP(\mathcal{P}') \models m(Z', w, Y')$ by $(m1)$. In the following three cases, a type 2, 3, or 4 statement $A.r \longleftarrow e$ of $\mathcal{P}''$ is used; such a statement must be mapped from a type 2, 3, 4 statement in $\mathcal{P}'$. Since all such statements in $\mathcal{P}'$ are also in $\mathcal{P}$ and do not change in the mapping, $A.r \longleftarrow e \in \mathcal{P} \cap \mathcal{P}'$. Case $(m2)$: $Z.w \longleftarrow Z_1.w_1 \in \mathcal{P}'', \mathcal{P}, \mathcal{P}'$ and $m(Z_1, w_1, Y) \in T_{SP(\mathcal{P}'')}\uparrow^i$. From induction hypothesis, $SP(\mathcal{P}') \models m(Z_1', w_1, Y')$ and $[Z_1'] = Z_1$ and $[Y'] = Y$. Because $Z_1 \in \mathsf{Principals}(\mathcal{P})$, it must be that $Z_1' = Z_1$. The conclusion follows from $(m2)$. Case $(m3)$: $Z.w \longleftarrow Z.w_1.w_2 \in \mathcal{P}'', \mathcal{P}, \mathcal{P}'$ and $m(Z, w_1, F), m(F, w_2, Y) \in T_{SP(\mathcal{P}'')}\uparrow^i$ for some principal F. By induction hypothesis, $SP(\mathcal{P}') \models m(Z, w_1, F'), m(F'', w_2, Y')$ and $[F'] = [F''] = F$. Since $Z.w_1 \in \mathsf{SigRoles}(\mathcal{P}, \mathcal{P}', \mathcal{Q})$, by definition of $\equiv$ applied to $F' \equiv F''$, $SP(\mathcal{P}') \models m(Z, w_1, F'')$. The claim follows from $(m3)$. Case $(m4)$: $Z.w \longleftarrow Z_1.w_1 \cap Z_2.w_2 \in \mathcal{P}'', \mathcal{P}, \mathcal{P}'$ and $m(Z_1, w_1, Y), m(Z_2, w_2, Y) \in T_{SP(\mathcal{P}'')}\uparrow^i$. By induction hypothesis and the fact $Z_1, Z_2 \in \mathsf{Principals}(\mathcal{P})$, $SP(\mathcal{P}') \models m(Z_1, w_1, Y'), m(Z_2, w_2, Y'')$ and $[Y'] = [Y''] = Y$. By definition of $\equiv$, $SP(\mathcal{P}') \models m(Z_2, w_2, Y')$. Therefore, $SP(\mathcal{P}') \models m(Z, w, Y')$. ∎

Observe that in the proof, only roles in the body of type-3 and 4 statements need to be collected. This may be used to explain why containment in *BRT* is efficiently decidable.

References

[1] Martín Abadi. On SDSI's linked local name spaces. *Journal of Computer Security*, 6(1–2):3–21, 1998.

[2] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote trust-management system, version 2. IETF RFC 2704, September 1999.

[3] Ajay Chander, Drew Dean, and John C. Mitchell. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 27–43, June 2001.

[4] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[5] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[6] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. IETF RFC 2693, September 1999.

[7] Michael R. Garey and David J. Johnson. *Computers And Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[8] Joseph Halpern and Ron van der Meyden. A logic for SDSI's linked local name spaces. *Journal of Computer Security*, 9(1,2):47–74, 2001.

[9] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[10] Somesh Jha and Thomas Reps. Analysis of SPKI/SDSI certificates using model checking. In *Proceedings of the 15th Computer Security Foundations Workshop*, 2002.

[11] Butler W. Lampson. Protection. In *Proceedings 5th Princeton Conference on Information Sciences and Systems*, 1971. Reprinted in ACM Operating Systems Rev. 8, 1 (Jan. 1974), pp 18-24.

[12] Ninghui Li. Local names in SPKI/SDSI. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 2–15. IEEE Computer Society Press, July 2000.

[13] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages*, January 2003. To appear.

[14] Ninghui Li, John C. Mitchell, and William 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.

[15] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*. To appear.

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

[17] John W. Lloyd. *Foundations of Logic Programming, Second Edition*. Springer, 1987.

[18] Ravi S. Sandhu. The Schematic Protection Model: Its definition and analysis for acyclic attenuating systems. *Journal of ACM*, 35(2):404–432, 1988.

[19] Ravi S. Sandhu. The typed access matrix model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 122–136. IEEE Computer Society Press, 1992.