Beyond Proof-of-compliance: Security Analysis in Trust Management

NINGHUI LI
Purdue University
JOHN C. MITCHELL
Stanford University
and
WILLIAM H. WINSBOROUGH
George Mason University

Trust management is a form of distributed access control that allows one principal to delegate some access decisions to other principals. While the use of delegation greatly enhances flexibility and scalability, it may also reduce the control that a principal has over the resources it owns. Security analysis asks whether safety, availability, and other properties can be maintained while delegating to partially trusted principals. We show that in contrast to the undecidability of classical Harrison-Ruzzo-Ullman safety properties, our primary security properties are decidable. In particular, most security properties we study are decidable in polynomial time. The computational complexity of containment analysis, the most complicated security property we study, forms a complexity hierarchy based on the expressive power of the trust management language.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Complexity of proof procedures*

General Terms: Security, Theory, Languages

Additional Key Words and Phrases: access control, trust management, distributed system security, logic programs

1. INTRODUCTION

Access control techniques, which govern whether one party can access resources and objects controlled by another party, are useful in protecting the confidentiality, integrity, and availability of information. Traditional access control schemes make authorization decisions based on the identity of the requester. However, in decentralized or multicentric environments, the resource owner and the requester often are unknown to one another, making access control based on identity ineffective. For example, although a certificate authority may assert that the requester's name is John Q. Smith, if this name is unknown to

A preliminary version of this paper appeared in *Proceedings of 2003 IEEE Symposium on Security and Privacy* under the title "Beyond proof-of-compliance: Safety and availability analysis in trust management". Most of this work was performed while the first author was at the Department of Computer Science, Stanford University in Stanford, CA 94305, and the third author was at Network Associates Laboratories in Rockville, MD 20850.
Authors' addresses: Ninghui Li, Department of Computer Sciences, Purdue University, 656 Oval Drive, West Lafayette, IN 47907-2086, USA; email: `ninghui@cs.purdue.edu`. John C. Mitchell, Department of Computer Science, Gates 4B, Stanford, CA 94305-9045, USA; email: `mitchell@cs.stanford.edu`. William H. Winsborough, Center for Secure Information Systems, George Mason University, Fairfax, VA 22030-4444, USA; email: `wwinsborough@acm.org`.

the access mediator, the name itself does not aid in making an authorization decision. What is needed is information about the rights, qualifications, responsibilities and other characteristics assigned to John Q. Smith by one or more authorities, as well as trust information about the authorities themselves.

Trust management [Blaze et al. 1996; Blaze et al. 1999a; 1999b; Rivest and Lampson 1996; Ellison et al. 1999; Clarke et al. 2001; Gunter and Jim 2000; Jim 2001; Li et al. 2003; Li et al. 2003; Li et al. 2002; Li and Mitchell 2003a; Weeks 2001] is an approach to access control in decentralized distributed systems with access control decisions based on policy statements made by multiple principals. In trust management systems, statements that are maintained in a distributed manner are often digitally signed to ensure their authenticity and integrity; such statements are called *credentials* or *certificates*. A key aspect of trust management is delegation: a principal may transfer limited authority over one or more resources to other principals. While the use of delegation greatly enhances flexibility and scalability, it may also reduce the control that a principal has over the resources it owns. As delegation gives a certain degree of control to a principal that may be only partially trusted, a natural security concern is whether a resource owner nonetheless has some guarantees about who can access their resources. If we think of the union of all policies of all principals as the state of a trust management (TM) system, then a state may change as the result of a single step that adds or removes a policy statement, or as the result of a finite sequence of such steps. A resource owner generally has control over some part of the state, but cannot control all possible changes. In this paper, we consider the *security analysis* problem, which asks what accesses may be allowed or prevented by prospective changes in the state of a TM system.

A few definitions are useful for stating the security analysis problem more precisely. In general, a TM language has a syntax for specifying *policy statements* and *queries*, together with an entailment relation $\vdash$. We call a set $\mathcal{P}$ of policy statements a *state* of a TM system. Given a state $\mathcal{P}$ and a query $\mathcal{Q}$, the relation $\mathcal{P} \vdash \mathcal{Q}$ means that $\mathcal{Q}$ is true in $\mathcal{P}$. When $\mathcal{Q}$ arises from an access request, $\mathcal{P} \vdash \mathcal{Q}$ means that access $\mathcal{Q}$ is allowed in $\mathcal{P}$; a proof demonstrating $\mathcal{P} \vdash \mathcal{Q}$ is then called a *proof-of-compliance*.

Recognizing that a principal or a coalition of cooperating principals may control only a part of the global state, we assume there is a *restriction rule*, $\mathcal{R}$, that defines how states may be changed. For example, the principal in question may consider the part of the state controlled by fully trusted principals to be fixed, while considering that other principals may remove some policy statements and/or add new ones. Given a state $\mathcal{P}$ and a restriction rule $\mathcal{R}$, we write $\mathcal{P} \mapsto_{\mathcal{R}} \mathcal{P}'$ if the change from $\mathcal{P}$ to $\mathcal{P}'$ is allowed by $\mathcal{R}$, and $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ if a sequence of zero or more allowed changes leads from $\mathcal{P}$ to $\mathcal{P}'$. If $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, we say that $\mathcal{P}'$ is *$\mathcal{R}$-reachable* from P, or simply $\mathcal{P}'$ is *reachable*, when $\mathcal{P}$ and $\mathcal{R}$ are clear from context.

DEFINITION 1. Let $\mathcal{P}$ be a state, $\mathcal{R}$ a restriction rule, and $\mathcal{Q}$ a query. *Existential security analysis* takes the form: Does there exist $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$ and $\mathcal{P}' \vdash \mathcal{Q}$? When the answer is affirmative, we say $\mathcal{Q}$ is *possible* given $\mathcal{P}$ and $\mathcal{R}$. *Universal security analysis* takes the form: For every $\mathcal{P}'$ such that $\mathcal{P} \overset{*}{\mapsto}_{\mathcal{R}} \mathcal{P}'$, does $\mathcal{P}' \vdash \mathcal{Q}$? If so, we say $\mathcal{Q}$ is *necessary* given $\mathcal{P}$ and $\mathcal{R}$.

Here are some motivating examples of security analysis problems.

Simple Safety. (Existential) Does there exist a reachable state in which a specific (pre-

sumably untrusted) principal has access to a given resource?

Simple Availability. (Universal) In every reachable state, does a specific (presumably trusted) principal have access to a given resource?

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

Liveness. (Existential) Does there exist a reachable state in which no principal has access to a given resource?

Mutual Exclusion. (Universal) In every reachable state, are two given properties (or two given resources) mutually exclusive, i.e., no principal has both properties (or access to both resources) at the same time?

Containment. (Universal) In every reachable state, does every principal that has one property (e.g., has access to a resource) also have another property (e.g., is an employee)? Containment can express safety or availability (e.g., by interchanging the two example properties in the previous sentence).

Simple safety analysis was first formalized by Harrison et al. [Harrison et al. 1976] in the context of the well-known access matrix model [Lampson 1971; Graham and Denning 1972]. Simple safety analysis was referred to as *safety analysis* because other analysis problems were not considered. The model in [Harrison et al. 1976] is commonly known as the HRU model. In the general HRU model, *safety analysis* is undecidable [Harrison et al. 1976]. A number of protection models were developed to make safety analysis more tractable. Lipton and Snyder introduced the take-grant model [Lipton and Snyder 1977], in which simple safety can be decided in linear time. Sandhu introduced the Schematic Protection Model [Sandhu 1988], and the Typed Access Matrix model [Sandhu 1992]. In these previous works, only simple safety analysis are considered; the other kinds of analysis listed above were not. As some of the analysis problems are about properties other than safety (e.g., availability), we use the term *security analysis* rather than safety analysis.

To the best of our knowledge, security analysis for TM systems has not been investigated previously as such. In this paper, we define a precise model for security analysis in trust management. The policy languages we consider are languages in the RT family of Role-based Trust-management languages [Li et al. 2003; Li et al. 2002; Li and Mitchell 2003a]. The RT family combines the strengths of Role-Based Access Control (RBAC) [Sandhu et al. 1996] and previous trust-management (TM) systems. Semantics for the RT family is defined by translating each statement into a logic programming clause. In this paper, we consider four languages in the RT family; they are denoted by $\mathsf{RT}[\,]$, $\mathsf{RT}[\cap]$, $\mathsf{RT}[\twoheadleftarrow]$, and $\mathsf{RT}[\twoheadleftarrow, \cap]$. $\mathsf{RT}[\,]$ is the most basic language in the family; it has two types of statements: *simple member* and *simple inclusion*. $\mathsf{RT}[\cap]$ adds to $\mathsf{RT}[\,]$ *intersection inclusion* statements. $\mathsf{RT}[\twoheadleftarrow]$ adds to $\mathsf{RT}[\,]$ *linking inclusion* statements, which can be used to express attribute-based delegation. $\mathsf{RT}[\twoheadleftarrow, \cap]$ has both intersection inclusion and linking inclusion; $\mathsf{RT}[\twoheadleftarrow, \cap]$ is a slightly simplified (yet expressively equivalent) version of the RT_0 language described in [Li et al. 2003].

All the security analysis problems listed above are considered. While the TM language we are studying supports delegation and the kinds of analysis problems we consider are more general, somewhat surprisingly, these problems are decidable. Simple safety, simple availability, bounded safety, liveness, and mutual exclusion analysis for $\mathsf{RT}[\twoheadleftarrow, \cap]$ (and

hence for the other three sub-languages of $\mathsf{RT}[\twoheadleftarrow, \cap]$) can all be answered in time polynomial in the size of the state $\mathcal{P}$. These analysis problems are answered by evaluating queries against logic programs derived from the state $\mathcal{P}$ and the restriction rule $\mathcal{R}$.

Containment analysis is the most interesting case, both in terms of usefulness and in terms of technical challenge. The computational complexity of containment analysis depends on the language features. In $\mathsf{RT}[\,]$, the most basic language, containment analysis is in $\mathbf{P}$. Containment analysis become more complex when additional policy language features are used. Containment analysis is $\mathbf{coNP}$-complete for $\mathsf{RT}[\cap]$, $\mathbf{PSPACE}$-complete for $\mathsf{RT}[\twoheadleftarrow]$, and decidable in $\mathbf{coNEXP}$ for $\mathsf{RT}[\twoheadleftarrow, \cap]$. These complexity properties are proved using techniques and results from logic programming, formal languages, and automata theory. For $\mathsf{RT}[\,]$, we use logic programs derived from $\mathcal{P}$ and $\mathcal{R}$ to perform containment analysis. These logic programs use negation-as-failure in a stratified manner [Apt et al. 1988]. For $\mathsf{RT}[\cap]$, we show that containment analysis is essentially equivalent to determining validity of propositional logic formulas. The $\mathsf{RT}[\twoheadleftarrow]$ language is expressively equivalent to SDSI (Simple Distributed Security Infrastructure) [Rivest and Lampson 1996; Clarke et al. 2001], and is related to a class of string rewriting systems modelled using pushdown systems [Bouajjani et al. 1997]. We show that containment analysis in $\mathsf{RT}[\twoheadleftarrow]$ can be reduced to determining containment among reachable configurations of pushdown systems, which is again reduced to determining containment of languages accepted by Nondeterministic Finite Automata (NFAs). For the case of $\mathsf{RT}[\twoheadleftarrow, \cap]$, we show that if a containment does not hold, then there must exist a counter-example state (i.e., a reachable state in which the containment does not hold) of size at most exponential in the size of the input.

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, liveness, and mutual exclusion. In Section 4, we present results about containment analysis. We discuss related work in Section 5, and conclude in Section 6. An appendix contains proofs that are not included in the main body.

2. A CONCRETE SECURITY ANALYSIS PROBLEM

The abstract definition of security analysis in Definition 1 has three parameters: the language used to express the state $\mathcal{P}$, the form of query $\mathcal{Q}$, and the form of restriction rule $\mathcal{R}$. In this section, we define concrete security analysis problems by supplying these parameters. We give the syntax of the language for specifying policy statements in Section 2.1 and the semantics for the language in Section 2.2. We present the formulation of queries in Section 2.3 and the restriction rules in Section 2.4. In Section 2.5, we explain our query formulation in light of how restriction rules are defined. Finally, in Section 2.6 we discuss how security analysis can be used to achieve security objectives.

2.1 Syntax of The TM Language

The policy languages we consider are in the RT family of Role-based Trust-management languages [Li et al. 2002]. More specifically, we consider $\mathsf{RT}[\twoheadleftarrow, \cap]$ and its three sub-languages: $\mathsf{RT}[\,]$, $\mathsf{RT}[\twoheadleftarrow]$, and $\mathsf{RT}[\cap]$. The basic constructs of $\mathsf{RT}[\twoheadleftarrow, \cap]$ are *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 a word over some given standard alphabet. 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 that are members of this role. Each principal A has the authority to designate the members of each role of the form $A.r$. An access control permission is represented as a role as well; for example, that B is a member of the role of $A.r$ may represent that B has the permission to do action r on the object A.

There are four types of policy statements in $\mathsf{RT}[\twoheadleftarrow, \cap]$, each corresponding to a different way of defining role membership. Each statement has the form $A.r \longleftarrow e$, where $A.r$ is a role and e is a role expression, to be defined below. We read "$\longleftarrow$" as "includes", and say the policy statement *defines* the role $A.r$.

— *Simple Member*: $A.r \longleftarrow D$
This statement means that A asserts that D is a member of A's r role.

— *Simple Inclusion*: $A.r \longleftarrow B.r_1$
This statement means that A asserts that its r role includes (all members of) B's r_1 role. This represents a delegation from A to B, as B may add principals to become members of the role $A.r$ by issuing statements defining $B.r_1$.

— *Linking Inclusion*: $A.r \longleftarrow A.r_1.r_2$
We call $A.r_1.r_2$ a *linked role*. This statement means that A asserts that $A.r$ includes $B.r_2$ for every B that is a member of $A.r_1$. This represents a delegation from A to all the members of the role $A.r_1$.

— *Intersection Inclusion*: $A.r \longleftarrow B_1.r_1 \cap B_2.r_2$
We call $B_1.r_1 \cap B_2.r_2$ an *intersection*. This statement means that A asserts that $A.r$ includes every principal who is a member of both $B_1.r_1$ and $B_2.r_2$. This represents partial delegations from A to B_1 and to B_2.

A *role expression* is a principal, a role, a linked role, or an intersection. 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 $\mathsf{Roles}(\mathcal{P}) = \{\mathsf{A.r} \mid \mathsf{A} \in \mathsf{Principals}(\mathcal{P}), \mathsf{r} \in \mathsf{Names}(\mathcal{P})\}$. $\mathsf{RT}[\twoheadleftarrow, \cap]$ is a slightly simplified (yet expressively equivalent) version of RT_0 [Li et al. 2003]. [1]

In this paper, we consider also the following sub-languages of $\mathsf{RT}[\twoheadleftarrow, \cap]$: $\mathsf{RT}[\,]$ has only simple member and simple inclusion statements, $\mathsf{RT}[\twoheadleftarrow]$ adds to $\mathsf{RT}[\,]$ linking inclusion statements, and $\mathsf{RT}[\cap]$ adds to $\mathsf{RT}[\,]$ intersection inclusion statements.

EXAMPLE 1. An example that uses the four types of statements is given in Figure 1.

The four types of statements in $\mathsf{RT}[\twoheadleftarrow, \cap]$ cover the most common delegation relationships in other TM languages such as SPKI/SDSI [Ellison et al. 1999; Clarke et al. 2001] and KeyNote [Blaze et al. 1999a]. The sub-language $\mathsf{RT}[\twoheadleftarrow]$ can be viewed as a simplified yet expressively equivalent version of SDSI. SDSI allows long linked names, which correspond to expressions of the form $A.r_1.r_2.r_2.\cdots.r_n$. As observed in [Li et al. 2003], such expressions can be broken up by introducing intermediate roles and additional statements. With the exception of thresholds, the delegation relationships (though, not the S-expression-based representation of permission) in SPKI's 5-tuples, can be captured by using simple member statements and a restricted form of simple inclusion statements. A

[1] $\mathsf{RT}[\twoheadleftarrow, \cap]$ simplifies RT_0 in that intersection inclusion statements in $\mathsf{RT}[\twoheadleftarrow, \cap]$ allow the intersection of only two roles; in RT_0, the intersection may contain k components, each of which can be a principal, a role, or a linked role. RT_0 statements using such intersections can be equivalently expressed in $\mathsf{RT}[\twoheadleftarrow, \cap]$ by introducing intermediate roles and additional statements. This helps simplify the proofs in this paper.

The state $\mathcal{P}$ consists of the following statements:

SA.access $\longleftarrow$ SA.manager	(1)
SA.access $\longleftarrow$ SA.delegatedAccess $\cap$ HR.employee	(2)
SA.manager $\longleftarrow$ HR.manager	(3)
SA.delegatedAccess $\longleftarrow$ SA.manager.access	(4)
HR.employee $\longleftarrow$ HR.manager	(5)
HR.employee $\longleftarrow$ HR.programmer	(6)
HR.manager $\longleftarrow$ Alice	(7)
HR.programmer $\longleftarrow$ Bob	(8)
HR.programmer $\longleftarrow$ Carl	(9)
Alice.access $\longleftarrow$ Bob	(10)

Given the state $\mathcal{P}$ above, we have.

$$
\begin{aligned}
\mathsf{Principals}(\mathcal{P}) &= \{\text{SA, HR, Alice, Bob, Carl}\} \\
\mathsf{Names}(\mathcal{P}) &= \{\text{access, manager, delegatedAccess, employee, programmer}\} \\
\mathsf{Roles}(\mathcal{P}) &= \{A.r \mid A \in \mathsf{Principals}(\mathcal{P}), \mathsf{r} \in \mathsf{Names}(\mathcal{P})\} \\
&= \{\text{SA.access, SA.manager, } \cdots, \text{ SA.programmer, HR.access, } \cdots, \text{ Carl.programmer}\}
\end{aligned}
$$

Fig. 1. An example of a state $\mathcal{P}$ in $\mathsf{RT}[\twoheadleftarrow, \cap]$. The system administrator 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 defining employees and managers.

SPKI 5-tuple in which A delegates a permission r to B can be represented as $A.r \longleftarrow B$. A SPKI 5-tuple in which A delegates r to B and allows B to further delegate r can be represented as two $\mathsf{RT}[\twoheadleftarrow, \cap]$ statements: $A.r \longleftarrow B$ and $A.r \longleftarrow B.r$. Similar analogies can be drawn for KeyNote [Blaze et al. 1999a].

SPKI/SDSI does not have intersection inclusion statements but allows threshold subjects in 5-tuples. Using threshold subjects in SPKI/SDSI, one can express a policy that grants a permission to a principal if k or more principals from a list of n principals grant the permission to the principal. The capabilities of the intersection operator in $\mathsf{RT}[\twoheadleftarrow, \cap]$ and the threshold subjects in SPKI/SDSI are largely incomparable. One cannot express in SPKI/SDSI a policy that grants a permission to any principal who has both attribute r_1 and attribute r_2, a policy easily expressed using intersection. On the other hand, one cannot use intersection statements to achieve the effects of threshold subjects either. In the RT family, the functionalities of threshold subjects are achieved using manifold roles and two new kinds of statements, which are introduced in [Li et al. 2002]. Security analysis for RT languages with these additional features is beyond the scope of this paper. See [Li and Mitchell 2003b] for a more detailed comparison of SPKI/SDSI and the RT family of languages, .

Although $\mathsf{RT}[\twoheadleftarrow, \cap]$ is limited in that role names are constants, extending role names in $\mathsf{RT}[\twoheadleftarrow, \cap]$ to have parameterized roles does not change the nature of security analysis. The main techniques we use for security analysis in $\mathsf{RT}[\twoheadleftarrow, \cap]$ uses logic programs, which can be easily extended to handle parameterized roles. Therefore, we believe that many of the results and techniques developed for $\mathsf{RT}[\twoheadleftarrow, \cap]$ can be carried over to more expressive languages, e.g., RT_1 [Li et al. 2002], which adds to RT_0 the ability to have parameterized roles, RT_1^C [Li and Mitchell 2003a], which adds constraints to RT_1, and, to a certain extent, SPKI/SDSI and KeyNote.

The security analysis problem for $\mathsf{RT}[\twoheadleftarrow, \cap]$ involves new concepts and techniques. Semantics and inference for SDSI, which is essentially the sub-language $\mathsf{RT}[\twoheadleftarrow]$, has been

extensively studied [Abadi 1998; Clarke et al. 2001; Halpern and van der Meyden 2001; Jha and Reps 2002; Li 2000; Li et al. 2003]. Some of these studies consider only answering queries in a fixed state. Some consider universal analysis where no restriction is placed on how the state may grow [Abadi 1998; Halpern and van der Meyden 2001]. However, the most interesting aspect of security analysis — answering queries when restrictions are placed on state changes — has not been addressed in the previous studies.

2.2 Semantics of the TM Language

We give a formal characterization of the meaning of a set $\mathcal{P}$ of policy statements by translating each policy statement 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 $\mathcal{P}$.

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. Intuitively, $m(A, r, D)$ means that D is a member of the role $A.r$. $SP(\mathcal{P})$ is the set of all datalog clauses produced from policy statements in $\mathcal{P}$. The rules to generate the Semantic Program $SP(\mathcal{P})$ from $\mathcal{P}$ are shown below. Symbols that start with "?" represent logical variables.

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

A datalog program is a set of datalog clauses. Given a datalog program, $\mathcal{DP}$, its semantics can be defined through several equivalent approaches. The model-theoretic approach views $\mathcal{DP}$ 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 is equivalent to the set-theoretic semantics of RT_0 in [Li et al. 2003].

We now summarize a standard fixpoint characterization of the minimal Herbrand model, which we will use in the proofs in this paper. Given a datalog program $\mathcal{DP}$, let $\mathcal{DP}^{inst}$ be the ground instantiation of $\mathcal{DP}$ using constants in $\mathcal{DP}$; the *immediate consequence operator*, $T_{\mathcal{DP}}$, is defined as follows. Given a set K of ground logical atoms, $T_{\mathcal{DP}}(K)$ consists of all logical atoms, a, such that $a :- b_1, \ldots, b_n \in \mathcal{DP}^{inst}$, where $n \geq 0$, and either $n = 0$ or $b_j \in K$ for $1 \leq j \leq n$. The least fixpoint of $T_{\mathcal{DP}}$ is given by

$$T_{\mathcal{DP}}{\uparrow}^{\omega} = \bigcup_{i=0}^{\infty} T_{\mathcal{DP}}{\uparrow}^{i},\ \text{where } T_{\mathcal{DP}}{\uparrow}^{0} = \emptyset \text{ and } T_{\mathcal{DP}}{\uparrow}^{i+1} = T_{\mathcal{DP}}(T_{\mathcal{DP}}{\uparrow}^{i}), i \geq 0$$

The sequence $T_{\mathcal{DP}}{\uparrow}^{i}$ is an increasing sequence of subsets of a finite set. Thus, there exists an N such that $T_{\mathcal{DP}}(T_{\mathcal{DP}}{\uparrow}^{N}) = T_{\mathcal{DP}}{\uparrow}^{N}$. $T_{\mathcal{DP}}{\uparrow}^{\omega}$ is identical to the minimal Herbrand model of $\mathcal{DP}$ [Lloyd 1987]; therefore, $SP(\mathcal{P}) \models m(X, u, Z)$ if and only if $m(X, u, Z) \in T_{SP(\mathcal{P})}{\uparrow}^{\omega}$.

The semantic program, $SP(\mathcal{P})$, of the $\mathcal{P}$ given in Figure 1, is as follows.

$$
\begin{array}{ll}
m(\text{SA}, \text{access}, ?Z) :- m(\text{SA}, \text{manager}, ?Z) & (1)\\
m(\text{SA}, \text{access}, ?Z) :- m(\text{SA}, \text{delegatedAccess}, ?Z),\ m(\text{HR}, \text{employee}, ?Z) & (2)\\
m(\text{SA}, \text{manager}, ?Z) :- m(\text{HR}, \text{manager}, ?Z) & (3)\\
m(\text{SA}, \text{delegatedAccess}, ?Z) :- m(\text{SA}, \text{manager}, ?Y),\ m(?Y, \text{access}, ?Z) & (4)\\
m(\text{HR}, \text{employee}, ?Z) :- m(\text{HR}, \text{manager}, ?Z) & (5)\\
m(\text{HR}, \text{employee}, ?Z) :- m(\text{HR}, \text{programmer}, ?Z) & (6)\\
m(\text{HR},\ \text{manager},\ \text{Alice}) & (7)\\
m(\text{HR},\ \text{programmer},\ \text{Bob}) & (8)\\
m(\text{HR},\ \text{programmer},\ \text{Carl}) & (9)\\
m(\text{Alice},\ \text{access},\ \text{Bob}) & (10)
\end{array}
$$

The minimal Herbrand model of the above program has the following facts.

Iteration 0	$m(\text{HR},\ \text{manager},\ \text{Alice})$	$m(\text{HR},\ \text{programmer},\ \text{Bob})$
	$m(\text{HR},\ \text{programmer},\ \text{Carl})$	$m(\text{Alice},\ \text{access},\ \text{Bob})$
Iteration 1	$m(\text{SA},\ \text{manager},\ \text{Alice})$	$m(\text{HR},\ \text{employee},\ \text{Alice})$
	$m(\text{HR},\ \text{employee},\ \text{Bob})$	$m(\text{HR},\ \text{employee},\ \text{Carl})$
Iteration 2	$m(\text{SA},\ \text{delegatedAccess},\ \text{Carl})$	$m(\text{SA},\ \text{access},\ \text{Alice})$
Iteration 3	$m(\text{SA},\ \text{access},\ \text{Bob})$	

Fig. 2. The semantic program, $SP(\mathcal{P})$, of the $\mathcal{P}$ given in Figure 1 and the minimal Herbrand model of the program.

It has been shown that the minimal Herbrand model of $\mathcal{DP}$ can be computed in time linear in the size of $\mathcal{DP}^{inst}$ [Dowling and Gallier 1984]. If the total size of $\mathcal{DP}$ is M, then there are $O(M)$ constants in $\mathcal{DP}$. 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 $\mathcal{DP}^{inst}$ is $O(M^{v+1})$. As $|SP(\mathcal{P})| = O(|\mathcal{P}|)$ and each rule in $SP(\mathcal{P})$ has at most two variables, the worst-case complexity of evaluating $SP(\mathcal{P})$ is $O(|\mathcal{P}|^3)$.

EXAMPLE 2. The semantic program of the example in Figure 1 is given in Figure 2.

2.3 Queries

In this paper, we consider the following three forms of query $\mathcal{Q}$:

— *Membership*: $A.r \sqsupseteq \{D_1, \ldots, D_n\}$

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 $\{Z \mid SP(\mathcal{P}) \models m(A, r, Z)\} \supseteq \{D_1, \ldots, D_n\}$.

— *Boundedness*: $\{D_1, \ldots, D_n\} \sqsupseteq A.r$

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

— *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 $\{Z \mid SP(\mathcal{P}) \models m(X, u, Z)\} \supseteq \{Z \mid SP(\mathcal{P}) \models m(A, r, Z)\}$.

EXAMPLE 3. If $\mathcal{P}$ is the state given in Figure 1, the following queries yield the indi-

cated results:

Membership:	$\mathcal{P} \vdash$ SA.access $\sqsupseteq \{\text{Eve}\}$	(False)
Membership:	$\mathcal{P} \vdash$ SA.access $\sqsupseteq \{\text{Alice}\}$	(True)
Boundedness:	$\mathcal{P} \vdash \{\text{Alice}, \text{Bob}\} \sqsupseteq$ SA.access	(True)
Inclusion:	$\mathcal{P} \vdash$ HR.employee $\sqsupseteq$ SA.access	(True)

We consider alternate formulations for queries below after first considering what kinds of state change rules we consider in the analysis.

2.4 Restriction Rules on State Changes

Using statements in $\mathsf{RT}[\twoheadleftarrow, \cap]$, one can delegate control over resources to other principals. In Figure 1, the two statements SA.access$\longleftarrow$SA.delegatedAccess $\cap$ HR.employee and SA.delegatedAccess $\longleftarrow$ SA.manager.access together mean that any principal that is a manager can affect who can access the resource. For example, Alice could add Alice.access $\longleftarrow$ Carl giving Carl access. In the resulting state $\mathcal{P}'$, $\mathcal{P}' \vdash \{\text{Alice}, \text{Bob}\} \sqsupseteq$ SA.access is false, whereas the result is true for $\mathcal{P}$. From the System Administrator (SA)'s perspective, roles such as Alice.access are not under its control. New statements defining Alice.access may be issued by Alice and existing statements defining Alice.access may be revoked. In order for SA to understand the effect of the two statements mentioned above, SA may want to know whether some desirable security properties always hold even though statements defining roles such as Alice.access can be changed arbitrarily.

We now present a concrete formulation of restriction rules that enable one to articulate analysis questions concerning the states that are reachable based on changes to policy. To model 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*).

If a role $A.r$ that is g-restricted is defined to include a role $B.r_1$ that is g-unrestricted, then no new statement defining $A.r$ can be added; however, new statements defining $B.r_1$ can be added, indirectly adding new members to $A.r$.

An example of $\mathcal{R}$ is $(\emptyset, \mathsf{Roles}(\mathcal{P}))$, under which every role may grow without restriction, and no statement defining roles in $\mathsf{Roles}(\mathcal{P})$ can be removed. This models the case of having incomplete knowledge of a global policy state. In this case, one sees a set $\mathcal{P}$ of statements but thinks that there are other statements in the global state that are currently unknown, and one wants to know whether certain security properties always hold no matter what these unknown statements may be.

Another example is $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, where $\mathcal{G} = \mathcal{S} = \{X.u \mid X \in \{X_1, \ldots, X_k\}, u \in \mathsf{Names}(\mathcal{P})\}$. This corresponds to the scenario in which there are principals that are trusted and one wants to analyze the effect of policy changes of untrusted principals. Here $X_1, \ldots, X_k$ are identified as trusted, and other principals are not trusted.

If a principal X does not appear in the restriction rule $\mathcal{R}$, then for every role name r, by

definition $X.r$ is *g/s-unrestricted*. This models 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 may be g-unrestricted. We also allow a role to be g-restricted while being s-unrestricted. These generalizations provide more flexibility than simply identifying principals as either trusted or untrusted. This flexibility in practice helps reduce the number of times that security analysis needs to be performed. For example, if a security property holds when a role is g-unrestricted, then the property will continue to hold after adding new statements defining the role, so the analysis need not be repeated.

EXAMPLE 4. Referring again to the example in Figure 1, consider the restriction rule $\mathcal{R}$ given as follows:

$\mathcal{G}$ = { SA.access, SA.manager, SA.delegatedAccess, HR.employee }
$\mathcal{S}$ = { SA.access, SA.manager, SA.delegatedAccess, HR.employee, HR.manager }

In this restriction rule, SA and HR are assumed to be trusted; however $\mathcal{G}$ allows statements to be added defining HR.manager and HR.programmer. Thus adding such statements cannot invalidate any security property obtained by using $\mathcal{R}$.

Given the above $\mathcal{R}$, statements (1) to (7) cannot be removed, statements (8) to (10) may be removed, new statements defining roles in $\mathcal{G}$ cannot be added, and one can add new statements defining HR.manager, HR.programmer, Alice.access, Bob.access, Carl.access, etc. We now list some example analysis problem instances, together with the answers:

Simply safety analysis:	Is "SA.access $\sqsupseteq$ {Eve}" possible?	(Yes)
Simple availability analysis:	Is "SA.access $\sqsupseteq$ {Alice}" necessary?	(Yes)
Bounded safety analysis:	Is "{Alice, Bob} $\sqsupseteq$ SA.access" necessary.	(No)
Containment analysis:	Is "HR.employee $\sqsupseteq$ SA.access" necessary?	(Yes)

Observe that the availability property "SA.access $\sqsupseteq$ {Alice} is necessary" depends on HR.manager being s-restricted. Together with our observations above concerning repeated analysis, this illustrates the advantage of allowing $\mathcal{G} \neq \mathcal{S}$.

The restrictions we consider are static in the sense that whether or not a state change 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 Alternate Query Formulations

We now examine the way we formulate queries in Section 2.3 by considering some of the alternatives. A membership query $A.r \sqsupseteq \{D_1, \ldots, D_n\}$ can be translated to an inclusion query $A.r \sqsupseteq B.u$, in which $B.u$ is a new role, by adding $B.u \longleftarrow D_1, \ldots, B.u \longleftarrow D_n$ to $\mathcal{P}$ and making $B.u$ g/s-restricted. Similarly, boundedness queries can be translated to inclusion queries as well. We include membership and bounded queries nonetheless because they can be answered more efficiently than inclusion queries.

Each form of query can be generalized to allow compound role expressions that use linking and intersection. However, these generalized queries can be reduced to the forms above by adding new roles and statements to the policy. For instance, $\{\} \sqsupseteq A.r \cap A_1.r_1.r_2$ can be answered by adding $B.u_1 \longleftarrow A.r \cap B.u_2$, $B.u_2 \longleftarrow B.u_3.r_2$, and $B.u_3 \longleftarrow A_1.r_1$ to $\mathcal{P}$, in which $B.u_1$, $B.u_2$, and $B.u_3$ are new g/s-restricted roles, and by posing the query

$\{\} \sqsupseteq B.u_1$.

The three forms of queries can be varied to consider cardinality of roles rather than exact memberships. A cardinality variant of membership queries has the form "$|A.r| \geq n$", which means that the number of principals who are members of $A.r$ is no less than n. A cardinality variant of boundedness queries has the form "$n \geq |A.r|$". Cardinality variants of membership and boundedness queries can be answered similarly to the base queries. We do not consider a cardinality variant of inclusion queries in this paper.

2.6 Usage of Security Analysis

Security analysis can be used to help ensure that security requirements are met, and that they continue to be met after policy changes are made by autonomous, possibly malicious principals.

For the purposes of the current section, let us say that a query $\mathcal{Q}$, a restriction rule $\mathcal{R}$, and a *sign*, either $+$ or $-$, together formalize a *requirement*. For instance, one requirement might consider whether every who can access a particular confidential resource is an employee of the organization. In this case, the sign used would be $+$ to indicate that the condition should always hold, as this ensures that no one outside the organization can access the confidential resource. A policy $\mathcal{P}$ *complies with* a requirement $\langle \mathcal{Q}, \mathcal{R}, + \rangle$ if $\mathcal{Q}$ is necessary given $\mathcal{R}$ and $\mathcal{P}$, and $\mathcal{P}$ complies with $\langle \mathcal{Q}, \mathcal{R}, + \rangle$ if $\mathcal{Q}$ is necessary given $\mathcal{R}$ and $\mathcal{P}$.

An organization's System Security Officer (SSO) writes a set of requirements based on a restriction rule $\mathcal{R}$ that forbids changing roles that are under the control of trusted principals in the organization. Assuming that we start in a policy state that complies with all the requirements, security analysis ensures that this compliance can be preserved across changes to the policy state as long as principals identified as trusted in $\mathcal{R}$ cooperate as follows. When a change is made by a principal that is untrusted, it must be to a g/s-unrestricted role; such a change has been taken into account by the analysis and does not affect the compliance. When a change is made by a trusted principal but is allowed by $\mathcal{R}$, then nothing needs to be done, as such changes are taken into account by the analysis. When a change is made by a trusted principal and is not allowed by $\mathcal{R}$, i.e., adding a statement that defines a g-restricted role or removing a statement that defines a s-restricted role, the principal should perform security analysis to determine whether the security requirements are met for the state that would result from the prospective change and make the change only if the requirements are satisfied. Thus, the preservation of compliance does not depend on untrusted principals.

In the above usage, the SSO determines a set of requirements based on a single restriction rule. In general, other principals may specify requirements they wish to have maintained by the TM system. They may have differing sets of principals that they are willing to trust with running the analysis and preserving the requirements, which will consequently be using differing restriction rules.

It is significant that the usage pattern we are suggesting enables the enforcement of requirements that cannot be achieved by constructs in $\mathsf{RT}[\leftarrow, \cap]$, or most other trust management languages. This is because those languages are monotonic in the sense that adding statements to a policy cannot remove a principal from a role. By contrast, many of the requirements formalized above are non-monotonic, in the sense that adding statements to a policy that satisfies the requirement can yield a policy that does not. This is illustrated by the example of mutual exclusion of two roles. Monotonicity makes it impossible to

express within RT$[\twoheadleftarrow, \cap]$ that a principal cannot be added to both roles. However this is easily achieved by using security analysis as described above.

3. ANSWERING MEMBERSHIP AND BOUNDEDNESS QUERIES

RT$[\twoheadleftarrow, \cap]$ and its sub-languages are monotonic in the sense that more statements will derive more role membership facts. This follows from the fact that the semantic program is a positive logic program. This important monotonicity property allows us to derive efficient algorithms for membership and boundedness queries.

To answer a universal membership (simple availability) analysis instance that asks whether "$A.r \sqsupseteq \{D_1, \ldots, D_n\}$" is necessary given $\mathcal{P}$ and $\mathcal{R}$, one can consider the set of principals that are members of $A.r$ in every reachable state. We call this set the *lower-bound* of $A.r$. If the lower-bound of $A.r$ is a superset of $\{D_1, \ldots, D_n\}$, then the answer to the analysis is "yes"; otherwise, the answer is "no".

To compute the lower-bound of a role, consider the state obtained from $\mathcal{P}$ by removing all statements whose removal is permitted by $\mathcal{R}$. We denote this state by $\mathcal{P}|_{\mathcal{R}}$. Because $\mathcal{R}$ is static, the order of removing these statements does not matter, and $\mathcal{P}|_{\mathcal{R}}$ uniquely exists. Clearly, $\mathcal{P}|_{\mathcal{R}}$ is reachable; furthermore, $\mathcal{P}|_{\mathcal{R}} \subseteq \mathcal{P}'$ for every reachable $\mathcal{P}'$. As RT$[\twoheadleftarrow, \cap]$ is monotonic, the lower-bound of $A.r$ is the same as the set of principals who are members of the role $A.r$ in $\mathcal{P}|_{\mathcal{R}}$.

The lower-bound of $A.r$ can also be used to answer an existential boundedness (liveness) analysis that asks whether "$\{D_1, \ldots, D_n\} \sqsupseteq A.r$" is possible given $\mathcal{P}$ and $\mathcal{R}$. If the lower-bound of $A.r$ is a subset of $\{D_1, \ldots, D_n\}$, then the answer is "yes"; otherwise, the answer is "no".

Existential membership (simple safety) analysis and universal boundedness (bounded safety) analysis can be answered by computing an "upper-bound" of role memberships. The upper-bound of a role is the set of principals that could become a member of the role in some reachable state. Intuitively, such bounds can be computed by considering a "maximal reachable state". However, such a "state" may contain an infinite set of policy statements, and the upper-bounds of roles may be infinite. We will show that one can simulate the upper bounds by a finite set and derive correct answers.

In Section 3.1, we show how to compute the lower-bounds and how to use them to perform universal membership and existential boundedness analysis. In Section 3.2, we show how to simulate the upper-bounds and how to use them to perform existential membership and universal boundedness analysis.

3.1 The Lower-Bound

We now present the lower-bound program for a state $\mathcal{P}$ and a restriction $\mathcal{R}$; this program enables one to compute the lower-bounds of every role.

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

The worst-case complexity of evaluating the lower-bound program is $O(|\mathcal{P}|^3)$, as noted at the end of Section 2.2.

Observe that the above lower-bound program is essentially the same as the semantic program for the minimal state $\mathcal{P}|_{\mathcal{R}}$. 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 $SP(\mathcal{P}|_{\mathcal{R}}) \models m(A, r, D)$.*

PROOF. This follows directly from the observation stated before this fact. □

The following proposition asserts that the program $LB(\mathcal{P}, \mathcal{R})$ correctly computes the lower-bounds for every role $A.r$.

PROPOSITION 3.2. *$LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D)$ if and only if for every reachable $\mathcal{P}'$, $SP(\mathcal{P}') \models m(A, r, D)$.*

PROOF. The "only if" part: If $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D)$, then from Fact 3.1, $SP(\mathcal{P}|_{\mathcal{R}}) \models m(A, r, D)$. For every $\mathcal{P}'$ that is reachable, $\mathcal{P}|_{\mathcal{R}} \subseteq \mathcal{P}'$. Furthermore, the language $\mathsf{RT}[\leftarrow, \cap]$ is monotonic; therefore, $SP(\mathcal{P}') \models m(A, r, D)$.

The "if" part: if for every reachable $\mathcal{P}'$, $SP(\mathcal{P}') \models m(A, r, D)$, then $SP(\mathcal{P}|_{\mathcal{R}}) \models m(A, r, D)$, because $\mathcal{P}|_{\mathcal{R}}$ is reachable. From Fact 3.1, $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D)$. □

The methods to use the lower-bound program to answer universal membership analysis and existential boundedness analysis instances and the correctness of these methods are formally stated as the following two corollaries. Cardinality variants of these queries can be answered similarly.

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

PROOF. The "if" direction: If $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D_i)$ for every i such that $1 \leq i \leq n$, then by Proposition 3.2, $D_1, \ldots, D_n$ are members of $A.r$ in all reachable states, the query is therefore necessary.

The "only if" direction: if $A.r \supseteq \{D_1, \ldots, D_n\}$ is necessary, then for every reachable state $\mathcal{P}'$, $SP(\mathcal{P}') \models m(A, r, D_i)$ for every i such that $1 \leq i \leq n$. By Proposition 3.2, $LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, D_i)$ for every i such that $1 \leq i \leq n$. □

COROLLARY 3.4. *Given $\mathcal{P}$ and $\mathcal{R}$, a boundedness query $\{D_1, \ldots, D_n\} \supseteq A.r$ 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 if $\{D_1, \ldots, D_n\} \supseteq \{Z \mid LB(\mathcal{P}, \mathcal{R}) \models lb(A, r, Z)\}$, then there exists a reachable $\mathcal{P}'$ 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. □

EXAMPLE 5. Based on $\mathcal{P}$ given in Figure 1 and $\mathcal{R}$ given in Example 4, the state $\mathcal{P}|_{\mathcal{R}}$ consists of the following statements:

SA.access ⟵ SA.manager	(1)
SA.access ⟵ SA.delegatedAccess ∩ HR.employee	(2)
SA.manager ⟵ HR.manager	(3)
SA.delegatedAccess ⟵ SA.manager.access	(4)
HR.employee ⟵ HR.manager	(5)
HR.employee ⟵ HR.programmer	(6)
HR.manager ⟵ Alice	(7)

The minimal Herbrand model of $LB(\mathcal{P}, \mathcal{R})$ has the following facts:

$lb(\text{HR}, \text{manager}, \text{Alice})$ $lb(\text{HR}, \text{employee}, \text{Alice})$
$lb(\text{SA}, \text{manager}, \text{Alice})$ $lb(\text{SA}, \text{access}, \text{Alice})$

This enables us to determine that "SA.access $\sqsupseteq$ {Alice}" is necessary.

3.2 The Upper-Bound

The upper-bound of a role consists of all principals that could be a member of the role in some reachable state. One main difficulty in computing the upper-bounds of roles is that they may be infinite. We say that a role is *g-unbounded* if for every principal Z, there exists a reachable state $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A, r, Z)$. In other words, the upper-bound of $A.r$ contains every principal. A g-unrestricted role is clearly g-unbounded, as one can add a new statement to add an arbitrary principal to be a member of the role. A g-restricted role may also be g-unbounded, as it may (directly or indirectly) include a g-restricted role.

The following fact about g-unbounded roles says that one needs to consider only 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. This fact enables us to use one principal that does not already occur to be the representative of all new principals.

We now present the upper-bound program for a state $\mathcal{P}$ and a restriction rule $\mathcal{R}$. This program enables one to simulate the upper-bound of any role.

DEFINITION 4 THE UPPER-BOUND PROGRAM. Given $\mathcal{P}$ and $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, 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 rules $(u1)$ to $(u4)$ follow from the meanings of the four types of statements and are similar to the semantic program construction in Definition 2. The rule $(u0)$ means that for any role $A.r$ that is g-unrestricted, the upper-bound of $A.r$ contains every principal. It is incorrect to use $ub(A, r, \top)$ instead of $ub(A, r, ?Z)$ here, because given $B.r_1 \longleftarrow A.r \cap B.r_2$ and $B.r_2 \longleftarrow D$ where $A.r$ is g-unbounded and $B.r_2$ is g-restricted, we need to ensure that $ub(B, r_1, D)$ is true. The rule (u) means that for any role name r, the upper-bound of $\top.r$ contains every principal. This is so because $\top$ does not appear in $\mathcal{R}$, therefore, $\top.r$ is not g-restricted. The rule (u) is needed because given $A.r \longleftarrow A.r_1.r_2$, where $A.r$ is g-restricted and $A.r_1$ is g-unrestricted, we should ensure that the upper-bound of $A.r$ contains every principal.

The following proposition asserts that the upper-bound program correctly computes the upper-bounds of roles in $\mathsf{Roles}(\mathcal{P})$ when we restrict our attention to roles in $\mathsf{Principals}(\mathcal{P}) \cup \{\top\}$.

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 a reachable $\mathcal{P}'$ such that $SP(\mathcal{P}') \models m(A, r, Z)$.*

See Appendix A.1 for the proof. The following corollary shows that the upper-bound program can correctly tell whether a role is g-unbounded or not.

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

PROOF. Follows directly from Fact 3.5 and Proposition 3.6. □

The methods to use the upper-bound program to answer existential membership analysis and universal boundedness analysis and the correctness of these methods are stated in the following two corollaries. Cardinality variants of these queries can be answered similarly.

COROLLARY 3.8. *Given $\mathcal{P}$ and $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, a membership query $A.r \supseteq \{D_1, \ldots, D_n\}$ is possible if and only if at least 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. Consider two cases. Case one: $A.r \notin \mathsf{Roles}(\mathcal{P})$. In this case, conditions (2) and (3) will not hold, because no clause in $UB(\mathcal{P}, \mathcal{R})$ defines $A.r$. We now show that the query is possible if and only if condition (1), i.e., $A.r \notin \mathcal{G}$, holds. If $A.r \in \mathcal{G}$, then the

role $A.r$ will always be empty in every reachable state and the query will not be possible. If $A.r \notin \mathcal{G}$, then the role $A.r$ is g-unbounded; therefore, the query is possible.

Case two: $A.r \in \mathsf{Roles}(\mathcal{P})$. In this case, the first condition implies the second condition, following from Corollary 3.7. Condition (2) and condition (3) each implies that the query is possible. Therefore, if at least one of the three conditions hold, the query is possible. If none of the three conditions holds, then there exists D_j such that $1 \leq j \leq n$ and $UB(\mathcal{P}, \mathcal{R}) \not\models ub(A, r, D_j)$. If $D_j \in \mathsf{Principals}(\mathcal{P})$, then from Proposition 3.6, D_j is not a member of $A.r$ in any reachable state; therefore, the query is not possible. If $D_j \notin \mathsf{Principals}(\mathcal{P})$, then D_j is not a member of $A.r$ in any reachable state either. Because if D_j is a member of $A.r$ in some reachable state, then $A.r$ is g-unbounded according to Fact 3.5, and $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, \top)$ according to Corollary 3.7, which contradicts the assumption that Condition 2 does not hold. □

In Corollary 3.8, condition (1) is needed in addition to (2) to deal with the situation that either A or r does not occur in $\mathcal{P}$, in which case $A.r \notin \mathsf{Roles}(\mathcal{P})$ and $UB(\mathcal{P}, \mathcal{R})$ does not correctly compute the upper-bound of $A.r$. Condition (2) is needed in addition to (1) to deal with roles that are g-restricted, but g-unbounded. Condition (3) is needed to deal with the case that $A.r$ is not g-unbounded, but its upper-bound contains all principals in $\{D_1, \ldots, D_n\}$ nonetheless.

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

PROOF. Consider two cases. Case one: $A.r \notin \mathsf{Roles}(\mathcal{P})$. In this case, $\{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\} = \emptyset$; therefore, $\{D_1, \ldots, D_n\} \supseteq \{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\}$ always holds. If suffices to show that the query $\{D_1, \ldots, D_n\} \sqsupseteq A.r$ is necessary if and only if $A.r \in \mathcal{G}$. If $A.r \in \mathcal{G}$, then the role $A.r$ is empty in any reachable state; the query is thus necessary. If $A.r \notin \mathcal{G}$, then consider the state $\mathcal{P}' = \mathcal{P} \cup A.r \longleftarrow E$, where E is a principal not appearing in $\mathcal{P}$. The query is false in $\mathcal{P}'$.

Case two: $A.r \in \mathsf{Roles}(\mathcal{P})$. In this case, assume that $A.r \in \mathcal{G}$ and $\{D_1, \ldots, D_n\} \supseteq \{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\}$, we show that for any principal Z that is a member of $A.r$ in some reachable state $\mathcal{P}'$ (i.e., $SP(\mathcal{P}') \models m(A, r, Z)$), it must be that $Z \in \{D_1, \ldots, D_n\}$, thereby proving that the query is necessary. We first show that $Z \in \mathsf{Principals}(\mathcal{P})$; otherwise, it follows from Fact 3.5 that $A.r$ is g-unbounded, and it follows from Corollary 3.7 that $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, \top)$, which contradicts the assumption that $\{D_1, \ldots, D_n\} \supseteq \{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\}$. As $Z \in \mathsf{Principals}(\mathcal{P})$, it follows from Proposition 3.6 that $UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)$. From the assumption, $Z \in \{D_1, \ldots, D_n\}$.

If either $A.r \notin \mathcal{G}$ or $\{D_1, \ldots, D_n\} \not\supseteq \{Z | UB(\mathcal{P}, \mathcal{R}) \models ub(A, r, Z)\}$, then it is straightforward to show that the query is not necessary. □

EXAMPLE 6. Based on $\mathcal{P}$ given in Figure 1 and $\mathcal{R}$ given in Example 4, the minimal Herbrand model of $UB(\mathcal{P}, \mathcal{R})$ has the following facts, in which *principal* is Alice, Bob,

Carl or $\top$ and *role* is access, manager, delegatedAccess, employee, or programmer[2]:

$ub(\text{HR},\ \text{manager},\ \textit{principal})$ $ub(\text{HR},\ \text{programmer},\ \textit{principal})$
$ub(\text{Alice},\ \text{access},\ \textit{principal})$ $ub(\top, \textit{role},\ \textit{principal})$
$ub(\text{HR},\ \text{employee},\ \textit{principal})$ $ub(\text{SA},\ \text{delegatedAccess},\ \textit{principal})$
$ub(\text{SA},\ \text{manager},\ \textit{principal})$ $ub(\text{SA},\ \text{access},\ \textit{principal})$

This enables us to determine that "SA.access $\sqsupseteq$ {Eve}" is possible and "{Alice, Bob} $\sqsupseteq$ SA.access" is not necessary.

In the example, we see that the policy is not safe according to either the simple safety instance or the bounded safety instance. One reason is that the role HR.manager is g-unrestricted, meaning that new managers may be added. Another reason is that the role HR.programmer is g-unrestricted; therefore, new programmers may be added and access may be delegated to them. However, if the company knows that Eve is an enemy, then the company probably will not hire Eve as a manager or a programmer. In fact, simple safety is quite unnatural: to use it effectively, one has to be able to identify the principals that should never have access, the number of such principals could be arbitrary large. Bounded safety is also unnatural, one does not know, for example, who in the future the company will hire as a manager. A more natural policy is to ensure that, for example, only employees of the company are allowed to access the resource. This can be done by using inclusion queries.

3.3 Summary and Computational Complexities

The approaches that we have taken to answer membership queries utilize the facts that membership queries are monotonic, that is, given a membership query $\mathcal{Q}$, if $\mathcal{P} \vdash \mathcal{Q}$, then for every $\mathcal{P}'$ such that $\mathcal{P} \subseteq \mathcal{P}'$, $\mathcal{P}' \vdash \mathcal{Q}$. Because of this, universal membership queries can be answered by considered the minimal reachable state. If a membership query is true in the minimal state, then it is true in all states. Similarly, existential membership queries can be answered by considering the maximal reachable state.

Boundedness queries are anti-monotonic, that is, given a boundedness query $\mathcal{Q}$, if $\mathcal{P} \vdash \mathcal{Q}$, then for every $\mathcal{P}'$ such that $\mathcal{P}' \subseteq \mathcal{P}$, $\mathcal{P}' \vdash \mathcal{Q}$. Therefore, existential bounded queries can be answered by considering the minimal state. If the query is false in that state, then the query will be false in all other states. Similarly, universal bounded queries can be answered by considering the maximal reachable state.

As the lower-bound program is essentially the semantic program of $\mathcal{P}|_{\mathcal{R}} \subseteq \mathcal{P}$, the computational complexity for evaluating it is $O(|\mathcal{P}|^3)$.

As noted at the end of Section 2.2, the computational complexity for evaluating $UB(\mathcal{P}, \mathcal{R})$ is linear in the size of the ground instantiation of $UB(\mathcal{P}, \mathcal{R})$. There are $O(|\mathcal{P}|)$ rules in $UB(\mathcal{P}, \mathcal{R})$ corresponding to (u), $(u1)$, $(u2)$, $(u3)$, and $(u4)$, which have at most two variables per rule; therefore, the ground instantiation of these rules has total size $O(|\mathcal{P}|^3)$. There are $O(|\mathcal{P}|^2)$ instance rules of $(u0)$, because there are $O(|\mathcal{P}|)$ principals and $O(|\mathcal{P}|)$ role names in $\mathcal{P}$. However, each such rule has only one variable, and so the ground instantiation of these rules has size $O(|\mathcal{P}|^3)$. Therefore, the computational complexity for evaluating $UB(\mathcal{P}, \mathcal{R})$ is $O(|\mathcal{P}|^3)$.

[2] Actually, there are more of these, like $ub(\text{HR},\ \text{manager},\ \text{access})$, but when the third parameter is not a principal, these facts have no effect, so we elide them. Similarly, we elide the facts in which the second parameter is not a role name.

4. CONTAINMENT ANALYSIS: ANSWERING UNIVERSAL INCLUSION QUERIES

Inclusion queries are neither monotonic nor anti-monotonic. Given an inclusion 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 membership and boundedness queries is not applicable. We cannot simply look at a specific minimal (or maximal) state and answer the query.

In this paper, we restrict our attention to universal inclusion queries, as this is more interesting in terms of security properties than existential inclusion queries.

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. Observe that we use "contains" and "includes" as two technical terms that have different meanings.

We call the problem of determining whether a role contains another role the *containment analysis* problem. Note that "containment" is also used in information security to mean other things. Our definition of containment in this paper is specific to comparing memberships of two roles in all reachable states.

The problem of determining whether $X.u$ contains $A.r$ when one of $X.u$ and $A.r$ is not in $\mathsf{Roles}(\mathcal{P})$ can be answered easily as follows. If $A.r \notin \mathsf{Roles}(\mathcal{P})$, then either A or r does not occur in $\mathcal{P}$. When $A.r$ is g-restricted, $A.r$ will always be empty; therefore, $X.u$ contains $A.r$. When $A.r$ is g-unrestricted, then one may add arbitrary members to $A.r$, yet $A.r$ is not used to define any role, so $X.u$ does not contain $A.r$. In short, when $A.r \notin \mathsf{Roles}(\mathcal{P})$, $X.u$ contains $A.r$ if and only if $A.r$ is g-restricted. If $A.r \in \mathsf{Roles}(\mathcal{P})$ and $X.u \notin \mathsf{Roles}(\mathcal{P})$, 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 consider the case that both $X.u$ and $A.r$ are in $\mathsf{Roles}(\mathcal{P})$. As we show in this section, the computational complexity of containment analysis depends very much upon the delegation features of the langauge. In the four subsections in this section, we study containment analysis in $\mathsf{RT}[\,]$, $\mathsf{RT}[\cap]$, $\mathsf{RT}[\twoheadleftarrow]$, and $\mathsf{RT}[\twoheadleftarrow, \cap]$, respectively.

4.1 Containment Analysis in $\mathsf{RT}[\,]$ is in $\mathbf{P}$

Recall that the language $\mathsf{RT}[\,]$ has only simple member and simple inclusion statements. We now show that containment analysis in $\mathsf{RT}[\,]$ is in $\mathbf{P}$ by giving an efficient algorithm to perform containment analysis in $\mathsf{RT}[\,]$. This algorithm uses the logic program called the role-containment program.

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 statements. Forced containment can be computed by a method similar to that used for computing 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, as the member set of $A.r$ is empty in every reachable state. To compute this kind of containment, we observe that a g-restricted role $A.r$ 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:

EXAMPLE 7. $\mathcal{P} = \{A.r \longleftarrow A.r_1,\ A.r \longleftarrow D,\ A.r_1 \longleftarrow A.r,\ X.u \longleftarrow D\}$ and $\mathcal{R}$ is such that $\mathcal{G} = \{A.r, A.r_1\}$ and $\mathcal{S} = \{A.r, A.r_1, X.u\}$. In any $\mathcal{P}'$ that is $\mathcal{R}$-reachable from $\mathcal{P}$, the member sets of $A.r$ and $A.r_1$ are always $\{D\}$, and so both roles are contained by $X.u$.

A straightforward positive logic program cannot make the above inference. $X.u$ contains $A.r$ only if $X.u$ contains $A.r_1$ 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 principal E that is a member of $A.r$ in some state but not a member of $X.u$ in the same state.

DEFINITION 5. **(The Role Containment Program for RT[ ])** Given an RT[ ] 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 4-ary predicates: fc and nc. 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. (Recall that $\mathcal{P}|_{\mathcal{R}}$ is the minimal state that is $\mathcal{R}$-reachable from $\mathcal{P}$, obtained from $\mathcal{P}$ by removing all statements defining s-unrestricted roles; any statement in $\mathcal{P}|_{\mathcal{R}}$ exists in every reachable state.)

Add $fc(?X, ?u, ?X, ?u)$ (c)

For each $A.r \longleftarrow B.r_1$ in $\mathcal{P}|_{\mathcal{R}}$, add
$fc(A, r, ?Z, ?w) :- fc(B, r_1, ?Z, ?w)$ $(c1)$

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
  $nc(?X, ?u, A, r) :- \sim fc(?X, ?u, A, r), nc(?X, ?u, B, r_1)$ $(n2)$

Rule (c) says that every role is forced to contain itself. The intuition behind $(c1)$ is that, if $A.r \longleftarrow B.r_1$ exists in every reachable state, then $A.r$ is forced to contain $B.r_1$. The intuition behind $(n0)$ is that for $X.u$ to contain a g-unrestricted role $A.r$, $X.u$ has to be forced to contain $A.r$, because arbitrary new members may be added to $A.r$. The intuition behind $(n1)$ is that, as $A.r$ contains D, if $X.u$'s lower-bound does not contain D, then $X.u$ does not contain $A.r$ unless $X.u$ is forced to contain $A.r$. The "$\sim fc$" part is needed, because 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 but $X.u$ contains $A.r$ nontheless. 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 $\mathcal{DP}$, 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 stratified. Predicates in $BCP(\mathcal{P},\mathcal{R})$ are classified into two strata; the lower stratum has lb and fc, and the only predicate in the higher stratum is nc.

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

EXAMPLE 8. Consider again $\mathcal{P}$ and $\mathcal{R}$ of Example 7. $BCP(\mathcal{P},\mathcal{R})$ is given by the following:

$$\mathcal{DP}_1 \begin{cases} lb(A,r,?X) :- lb(A,r_1,?X) & (1)\\ lb(A,r,D) & (2)\\ lb(A,r_1,?X) :- lb(A,r,?X) & (3)\\ lb(X,u,D) & (4)\\ fc(A,r,?Z,?w) :- fc(A,r_1,?Z,?w) & (5)\\ fc(A,r_1,?Z,?w) :- fc(A,r,?Z,?w) & (6)\\ fc(?X,?u,?X,?u) & (7) \end{cases}$$

$$\mathcal{DP}_2 \begin{cases} nc(?X,?u,X,u) :- \sim fc(?X,?u,X,u) & (8)\\ nc(?X,?u,A,r) :- \sim fc(?X,?u,A,r), \sim lb(?X,?u,D) & (9)\\ nc(?X,?u,A,r) :- \sim fc(?X,?u,A,r), nc(?X,?u,A,r_1) & (10)\\ nc(?X,?u,A,r_1) :- \sim fc(?X,?u,A,r_1), nc(?X,?u,A,r) & (11) \end{cases}$$

The minimal Herbrand model of $\mathcal{DP}_1$ contains the following facts:

$$\begin{array}{lll} lb(A,r,D) & lb(A,r_1,D) & lb(X,u,D)\\ fc(A,r,A,r) & fc(A,r_1,A,r_1) & fc(X,u,X,u)\\ fc(A,r,A,r_1) & fc(A,r_1,A,r) & \end{array}$$

Using this to determine the truthfulness of negative literals in $\mathcal{DP}_2$, computing the minimal Herbrand model of $\mathcal{DP}_1 \cup \mathcal{DP}_2$ adds the following facts:

$$nc(A,r,X,u)\ \ nc(A,r_1,X,u)$$

The following lemma says that the fc predicate in $BCP(\mathcal{P},\mathcal{R})$ is always sound for role containment, i.e., if $fc(X,u,A,r)$ can be proved from $BCP(\mathcal{P},\mathcal{R})$, then $X.u$ contains $A.r$. Furthermore, if $A.r$ is g-unrestricted and $X.u$ contains $A.r$, then $fc(X,u,A,r)$ can be proved from $BCP(\mathcal{P},\mathcal{R})$. In other words, fc is complete when the second role is g-unrestricted.

LEMMA 4.1. *Given an* RT[ ] *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 RT[ ] can be answered by using the program $BCP(\mathcal{P},\mathcal{R})$.

PROPOSITION 4.2. *Given an* RT[ ] *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* $A.r$.

See Appendix A.2 for the proof.

Evaluating the semantics of a stratified program can be performed in time polynomial in the size of the program; therefore, containment analysis in RT[ ] is in $\mathbf{P}$.

4.2 Containment Analysis in RT[∩] is coNP-complete

RT[∩] adds to RT[ ] type-4 statements, which use the intersection operator. This makes the containment analysis problem in RT[∩] to become $\mathbf{coNP}$-complete. The intuition of the problem's $\mathbf{coNP}$-completeness is as follows. Determining whether $X.u$ contains a g-restricted role $A.r$ is the same as determining whether all the ways $A.r$ is defined are "contained" in the ways that $X.u$ is defined. In RT[∩], a role can be defined by multiple statements, which have the effect of disjunction, and a role can be defined using intersections, which have the effect of conjunction. As a result, one can use roles in RT[∩] to encode positive (only using conjunctions and disjunctions) formulas in propositional logic, and containment analysis subsumes the problem of determining validity of whether one positive propositional formula implies another such formula, a problem that is $\mathbf{coNP}$-complete.

Consider the following example: $\mathcal{P} = \{X.u \longleftarrow A.r_1 \cap A.r_2,\ A.r_1 \longleftarrow B.r_1,\ A.r_1 \longleftarrow B.r_2,\ A.r_2 \longleftarrow B.r_1,\ A.r_2 \longleftarrow B.r_3, A.r \longleftarrow B.r_2 \cap B.r_3\}$, $\mathcal{R}$ is such that $\mathcal{G} = \mathcal{S} = \{X.u, A.r_1, A.r_2\}$. Each role that is g-restricted can be written as a positive propositional formula where the g-unrestricted roles are used as propositional variables. In this example, $A.r_1$'s definition is $B.r_1 \vee B.r_2$, $A.r_2$'s definition is $B.r_1 \vee B.r_3$, $X.u$'s definition is $(B.r_1 \vee B.r_2) \wedge (B.r_1 \vee B.r_3)$, and $A.r$'s definition is $B.r_2 \wedge B.r_3$. Because $A.r$'s definition implies $X.u$'s definition, then any way one adds a principal to $A.r$ results in the same principal being added to $X.u$; therefore, $X.u$ contains $A.r$.

THEOREM 4.3. *Containment analysis in* RT[∩] *is* $\mathbf{coNP}$*-complete.*

See Appendix A.3 for the proof. The $\mathbf{coNP}$-hard part is by reducing the monotone 3SAT problem, which is $\mathbf{NP}$-complete, to the complement of containment analysis in RT[∩]. The reduction uses only g-unrestricted roles, and it is straightforward to change the proof to use only s-unrestricted roles. This shows that containment analysis in RT[∩] where all roles are g-restricted or where all roles are s-restricted is still $\mathbf{coNP}$-complete.

4.3 Containment Analysis in RT[←] is PSPACE-complete

RT[←] adds to RT[ ] type-4 statements, which use linked roles. This makes the containment analysis problem to go from $\mathbf{P}$ to $\mathbf{PSPACE}$-complete. The intuition is as follows. With the linking feature in RT[←], the ways in which a role is defined are encoded as a set of strings accepted by NFA's. The computational complexity of containment analysis in RT[←] is thus the same as the computational complexity to determine containment of languages accepted by two NFA's, which is $\mathbf{PSPACE}$-complete.

The main result of this section if the following theorem, which gives the computational complexity of containment analysis when we do not consider roles shrinking.

THEOREM 4.4. *Containment analysis in* RT[←] *where all roles in* Roles($\mathcal{P}$) *are s-restricted is* $\mathbf{PSPACE}$*-complete.*

This theorem will be proved in Sections 4.3.1, 4.3.2, and 4.3.3. Using Theorem 4.4, we can establish the exact complexity bound for containment analysis in $\mathsf{RT}[\twoheadleftarrow]$ in the general case.

THEOREM 4.5. *Containment analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *is* **PSPACE***-complete.*

PROOF. As the cases where all roles in $\mathsf{Roles}(\mathcal{P})$ are s-restricted are special cases of containment analysis in $\mathsf{RT}[\twoheadleftarrow]$, **PSPACE**-hardness follows immediately from Theorem 4.4.

We now show that the problem is in **PSPACE**. Given a containment analysis problem instance: $\mathsf{RT}[\twoheadleftarrow]$ state $\mathcal{P}$, a restriction rule $\mathcal{R} = (\mathcal{G}, \mathcal{S})$, and an inclusion query $\mathcal{Q}$, use the following algorithm. For each $\mathcal{P}'$ such that $\mathcal{P}' \subseteq \mathcal{P}$ and $\mathcal{P}'$ is $\mathcal{R}$-reachable from $\mathcal{P}$, perform containment analysis for $\mathcal{P}'$, $\mathcal{R}' = (\mathcal{G}, \mathsf{Roles}(\mathcal{P}))$, and $\mathcal{Q}$, reusing the space each time. The algorithm answers "no" if there exists a $\mathcal{P}'$ such that containment analysis answers no. Otherwise, the algorithm answers "yes".

If the containment does hold, then clearly this algorithm answers "yes". If the containment does not hold, then there exists a reachable state $\mathcal{P}_1$ and a principal E such that $SP(\mathcal{P}_1) \models m(A, r, E)$ and $SP(\mathcal{P}') \not\models m(X, u, E)$. Consider $\mathcal{P}_0 = \mathcal{P} \cap \mathcal{P}_1$, $\mathcal{P}_0$ is reachable; furthermore, $\mathcal{P}_1$ is $\mathcal{R}'$-reachable from $\mathcal{P}_0$; therefore, the algorithm answers "no".

From Theorem 4.4, we know that containment analysis with $\mathcal{P}'$ and $\mathcal{R}'$ takes space polynomial in $n = |\mathcal{P}'| + |\mathcal{R}'|$. As $n \leq |\mathcal{P}| + |\mathcal{R}|$, this algorithm takes space polynomial in the size of the original input. □

In the following, we prove Theorem 4.4. In Section 4.3.1, we introduce equivalence relationships between statements in $\mathsf{RT}[\twoheadleftarrow]$ and string rewriting systems; these relationships are useful in the proof. In Section 4.3.2, we prove that containment in $\mathsf{RT}[\twoheadleftarrow]$ without shrinkable roles is in **PSPACE** by showing that the problem can be reduced to checking containment of languages accepted by two NFA's. In Section 4.3.3, we prove that the problem is **PSPACE**-hard. In Section 4.3.4, we prove that containment in $\mathsf{RT}[\twoheadleftarrow]$ without growable roles is **coNP**-complete, establishing a tighter complexity bound for this special case of containment in $\mathsf{RT}[\twoheadleftarrow]$.

4.3.1 $\mathsf{RT}[\twoheadleftarrow]$ *as String Rewriting Systems.* Statements in $\mathsf{RT}[\twoheadleftarrow]$ can be viewed as rewriting rules. For example, the statement $A.r \longleftarrow B.r_1$ can be viewed as the rewriting rule $A\ r \mapsto B\ r_1$. The alphabet consists of all principals and role names. We consider rewriting over the set of *name strings*, i.e., strings that have the form of a principal followed by zero or more role names. When rewriting a name string ω using rewriting rules corresponding to statements in $\mathsf{RT}[\twoheadleftarrow]$, the rewriting occurs only on the left most, and the resulting string is also a name string.

DEFINITION 6 $\mathsf{RS}[\mathcal{P}]$. Given a set $\mathcal{P}$ of $\mathsf{RT}[\twoheadleftarrow]$ statements, let $\mathsf{RS}[\mathcal{P}]$ be the rewriting system generated by viewing each statement in $\mathcal{P}$ as a rewriting rule. Given two name strings ω_1 and ω_2, we write $\mathsf{RS}[\mathcal{P}] \rhd \omega_1 \rightarrowtail \omega_2$ if one can rewrite ω_1 into ω_2 in one step using a rewriting rule in $\mathsf{RS}[\mathcal{P}]$. We write $\mathsf{RS}[\mathcal{P}] \rhd \omega_1 \overset{*}{\rightarrowtail} \omega_2$ if using rewriting rules in $\mathsf{RS}[\mathcal{P}]$, one can rewrite ω_1 into ω_2 in zero or more steps.

We define $\mathsf{str}_{\mathcal{P}}(A.r)$ to denote the set $\{\omega \mid \mathsf{RS}[\mathcal{P}] \rhd A.r \overset{*}{\rightarrowtail} \omega\}$.

PROPOSITION 4.6. *Given a set* $\mathcal{P}$ *of* $\mathsf{RT}[\twoheadleftarrow]$ *statements,* $SP(\mathcal{P}) \models m(A, r, D)$ *if and only if* $\mathsf{RS}[\mathcal{P}] \rhd A\ r \overset{*}{\rightarrowtail} D$.

See Appendix A.4 for the proof.

RT[↞] is equivalent to SDSI [Clarke et al. 2001]. Jha and Reps [Jha and Reps 2002] pointed out that SDSI string rewriting systems correspond exactly to the class of string rewriting systems modelled using push-down systems [Bouajjani et al. 1997]. The same applies to the rewriting systems generated by RT[↞] statements.

Pushdown systems (PDSs) are similar to pushdown automata; however, unlike pushdown automata they do not have an input alphabet. Thus PDSs should not be viewed as language recognizers, but as mechanisms that specify possibly infinite-state transition systems.

A pushdown system is a triplet (Π, Γ, Δ), where Π is a finite set of states, Γ is a finite stack alphabet, and $\Delta \subseteq (\Pi \times \Gamma) \times (\Pi \times \Gamma^*)$ is a finite set of transition rules. If $((q, \gamma), (q', \omega)) \in \Delta$, then we write it as $\langle q, \gamma\rangle \hookrightarrow \langle q', \omega\rangle$.

A *configuration* of a PDS is a pair $\langle q, \omega\rangle$, where $q \in \Pi$ is a state and $\omega \in \Gamma^*$ represents the stack contents (from the top of the stack to the bottom of a stack). A configuration can be represented as a string $q \parallel \omega$ (q concatenated with ω).

We say that a configuration $\langle q, \omega_1\rangle$ can *directly reach* a configuration $\langle q', \omega_2\rangle$ if the following three conditions hold: (1) γ is the symbol on top of the stack ω_1, (2) there is a transition rule $\langle q, \gamma\rangle \hookrightarrow \langle q', \omega\rangle$, and (3) ω_2 is obtained from ω_1 by first poping γ and then pushing ω, i.e, $\omega_1 = \gamma\omega_1'$ and $\omega_2 = \omega\omega_1'$. We then define reachability among configurations in the straightforward manner.

Esparza et al. [Esparza et al. 2000] showed that given a PDS (Π, Γ, Δ) and a set C of configurations that is recognized by an NFA $\mathcal{A}$, the set of all configurations reachable from C is recognized by an NFA that has size polynomial in $n = |\Pi| + |\Gamma| + |\Delta| + |\mathcal{A}|$ and that can be constructed in time polynomial in n. This result is used in showing that containment analysis in RT[↞] is in **PSPACE**.

4.3.2 *Containment analysis in* RT[↞] *without shrinking is in* **PSPACE** . The key relationship between an instance of containment analysis and string rewriting lies in the following definition of characteristic sets.

DEFINITION 7 CHARACTERISTIC SET. Given an RT[↞] state $\mathcal{P}$, a restriction rule $\mathcal{R}$, and a role $A.r$, the characteristic set of $A.r$, denoted by $\chi_{\mathcal{P},\mathcal{R}}(A.r)$, is defined to be $\mathsf{str}_{\mathcal{P}}(A.r) \cap \kappa[\mathcal{P}, \mathcal{R}]$, where

$$\kappa[\mathcal{P}, \mathcal{R}] = \mathsf{Principals}(\mathcal{P}) \cup \{\mathsf{B\ r}\ \gamma \mid \mathsf{B.r} \text{ is g-unrestricted and } \gamma \in \mathsf{Names}(\mathcal{P})^*\}$$

The role $X.u$ contains the role $A.r$ if no matter how one adds a principal as a new member to $A.r$, $X.u$ also gets the principal as a member. Intuitively, each string in $\chi_{\mathcal{P},\mathcal{R}}(A.r)$ represents a distinct source of adding new members to the role $A.r$. Furthermore, the set $\chi_{\mathcal{P},\mathcal{R}}(A.r)$ describes all the ways of adding new members of the role $A.r$. Therefore, one can use characteristic sets to solve containment analysis.

LEMMA 4.7. *Given $\mathcal{P}$ and $\mathcal{R}$, $X.u$ contains $A.r$ if and only if $\chi_{\mathcal{P},\mathcal{R}}(X.u) \supseteq \chi_{\mathcal{P},\mathcal{R}}(A.r)$.*

PROOF. We first prove that if $\chi_{\mathcal{P},\mathcal{R}}(X.u) \not\supseteq \chi_{\mathcal{P},\mathcal{R}}(A.r)$ then $X.u$ does not contain $A.r$. If $\chi_{\mathcal{P},\mathcal{R}}(X.u) \not\supseteq \chi_{\mathcal{P},\mathcal{R}}(A.r)$, then there exists a string $\omega = B\ r_1 \cdots r_k$ such that $\omega \in \chi_{\mathcal{P},\mathcal{R}}(A.r)$ and $\omega \notin \chi_{\mathcal{P},\mathcal{R}}(X.u)$. Consider $\mathcal{P}' = \mathcal{P} \cup \{B.r_1 \longleftarrow C_1,\ C_1.r_2 \longleftarrow C_2,\ \cdots,\ C_{k-1}.r_k \longleftarrow C_k\}$, where $C_1, C_2, \cdots, C_k$ do not occur in $\mathcal{P}$

or $\mathcal{R}$. $\mathcal{P}'$ is a reachable state. $\mathsf{RS}[\mathcal{P}'] \rhd A\ r \overset{*}{\rightarrowtail} B\ r_1 \cdots r_k \overset{*}{\rightarrowtail} C_k$. From Proposition 4.6, $SP(\mathcal{P}') \models m(A, r, C_k)$. We now show that $SP(\mathcal{P}') \not\models m(X, u, C_k)$. Suppose, for the sake of contradiction, that $SP(\mathcal{P}') \models m(X, u, C_k)$. From Proposition 4.6, $\mathsf{RS}[\mathcal{P}'] \rhd X\ u \overset{*}{\rightarrowtail} C_k$. Consider the rewriting sequence, the rule applied in the last step has to be $C_{k-1}\ r_k \mapsto C_k$, because that is the only rule having a C_k on its right hand side. The rule applied in the second to last step has to be $C_{k-2}\ r_{k-1} \mapsto C_{k-1}$, because that is the only rule having C_{k-1} on its right-hand side, and so on. Therefore, the rewriting sequence must contain in its middle a sequence rewriting from $X\ u$ to $B\ r_1\ \cdots\ r_k$. Further observe that the rules in $\mathcal{P}'$ but not in $\mathcal{P}$ cannot be applied in this middle sequence. Therefore, $\mathsf{RS}[\mathcal{P}] \rhd X\ u \overset{*}{\rightarrowtail} \omega$. This contradicts the assumption that $\omega \notin \chi_{\mathcal{P},\mathcal{R}}(X.u)$.

We now prove that if $X.u$ does not contain $A.r$ then $\chi_{\mathcal{P},\mathcal{R}}(X.u) \not\supseteq \chi_{\mathcal{P},\mathcal{R}}(A.r)$. 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 Proposition 4.6, it follows that $\mathsf{RS}[\mathcal{P}'] \rhd A\ r \overset{*}{\rightarrowtail} E$. Consider the rewriting sequence that rewrites $A\ r$ into E; let the sequence be $A\ r \overset{*}{\rightarrowtail} \omega_1 \rightarrowtail \omega_2 \overset{*}{\rightarrowtail} E$, such that the step from ω_1 to ω_2 is the first step that uses a statement not in $\mathcal{P}$. Clearly, $\omega_1 \in \chi_{\mathcal{P},\mathcal{R}}(A.r)$ and $\omega_1 \notin \chi_{\mathcal{P},\mathcal{R}}(X.u)$. □

From the above lemma, we know that whether a role contains another can be determined by checking containment among characteristic sets.

PROPOSITION 4.8. *Containment analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *where all roles in* $\mathsf{Roles}(\mathcal{P})$ *are s-restricted is in* **PSPACE**.

PROOF. Given an $\mathsf{RT}[\twoheadleftarrow]$ state $\mathcal{P}$ and a restriction rule $\mathcal{R}$, to determine whether $X.u$ contains $A.r$, it suffices to check whether $\chi_{\mathcal{P},\mathcal{R}}(X.u) \supseteq \chi_{\mathcal{P},\mathcal{R}}(A.r)$. One can construct an NFA to recognize $\chi_{\mathcal{P},\mathcal{R}}(A.r)$ in time polynomial in $|\mathcal{P}| + |\mathcal{R}|$, and the NFA has size polynomial in $|\mathcal{P}| + |\mathcal{R}|$. This is because $\chi_{\mathcal{P},\mathcal{R}}(A.r) = \mathsf{str}_{\mathcal{P}}(A.r) \cap \kappa[\mathcal{P}, \mathcal{R}]$, and both $\mathsf{str}_{\mathcal{P}}(A.r)$ and $\kappa[\mathcal{P}, \mathcal{R}]$ are recognized by NFAs that can be constructed in time polynomial in $|\mathcal{P}| + |\mathcal{R}|$. Similarly, one can construct a polynomial size NFA to recognize $\chi_{\mathcal{P},\mathcal{R}}(X.u)$. It is known that determining containment of languages accepted by NFAs is in **PSPACE** [Hunt et al. 1976]. □

4.3.3 *Containment Analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *is* **PSPACE***-hard.* To prove that containment analysis in $\mathsf{RT}[\twoheadleftarrow]$ is **PSPACE**-hard, we use a reduction from the problem of checking containment among regular languages that are represented using NFAs. The problem is known to be **PSPACE**-complete, even when the alphabet has just two elements. (When regular languages are described using DFAs, the complexity is **NLOGSPACE**-complete.)

We consider regular languages over $\{0, 1\}$ that do not contain the empty string. They can be represented by a form of right linear grammars such that each production rule has one of the following two forms: $N_1 ::= N_2 b$, and $N_1 ::= b$, where N_1 and N_2 are nonterminals and $b \in \{0, 1\}$.

PROPOSITION 4.9. *Containment Analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *where all roles in* $\mathsf{Roles}(\mathcal{P})$ *are s-restricted is* **PSPACE***-hard.*

PROOF. Given two empty-string free regular languages over $\{0, 1\}$, L_1 and L_2, let G_1 and G_2 be the production rules for generating L_1 and L_2, and let S_1 and S_2 be the two start symbols. Wlog, assume that G_1 and G_2 do not share any non-terminal. Let $G = G_1 \cup G_2$.

For any nonterminal N in G, we write $L_G(N)$ to denote the language generated by G when using N as the start symbol.

We now reduce the problem of determining whether $L_G(S_1) \supseteq L_G(S_2)$ to a containment analysis problem. We use two principals A and B, two role names u_0 and u_1 for the two terminals 0 and 1, and one role name r_N for each nonterminal N. We define $\mathcal{P}_G$ to have the following statements. For each production $N ::= N'b$ in G, $\mathcal{P}_G$ has $A.r_N \longleftarrow A.r_{N'}.u_b$. For each production $N ::= b$, introduce a statement $A.r_N \longleftarrow B.u_b$. We define $\mathcal{R}_G$ to be such that all roles started with A are g/s-restricted, and all roles started with B are g-unrestricted.

It is not hard to see that a string $b_1 b_2 \cdots b_k \in L(N)$ if and only if $\mathsf{RS}[\mathcal{P}_G] \rhd A\ r_N \overset{*}{\rightarrowtail} B.u_{b_1}.u_{b_2}.\cdots.u_{b_k}$. One can further verify that there exists a one-to-one mapping from strings in $L_G(N)$ to $\chi_{\mathcal{P}_G,\mathcal{R}_G}(A.r_N)$ such that $L_G(N_1) \supseteq L_G(N_2)$ if and only if $\chi_{\mathcal{P}_G,\mathcal{R}_G}(A.r_{N_1}) \supseteq \chi_{\mathcal{P}_G,\mathcal{R}_G}(A.r_{N_2})$. From Lemma 4.7, it follows that $L_G(N_1) \supseteq L_G(N_2)$ if and only if $A.r_{N_1}$ contains $A.r_{N_2}$. □

4.3.4 *Containment analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *without growable roles.* Theorem 4.4 asserts that containment analysis in $\mathsf{RT}[\twoheadleftarrow]$ without shrinkable roles is **PSPACE**-complete. We now show that containment analysis in $\mathsf{RT}[\twoheadleftarrow]$ without growable roles is **coNP**-complete.

THEOREM 4.10. *Containment analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *where all roles in* $\mathsf{Roles}(\mathcal{P})$ *are g-restricted is* **coNP***-complete.*

See Appendix A.3 for the proof. The key observation used in the proof is that linked roles, similar to intersection, can simulate logical conjunction. Recall that the semantic rule for linking inclusion statements, $(m3)$, has a conjunction in the body, similar to that for intersection inclusion statements, $(m4)$.

4.4 Containment Analysis for $\mathsf{RT}[\twoheadleftarrow, \cap]$ is in coNEXP

We now study the computational complexity of containment analysis in $\mathsf{RT}[\twoheadleftarrow, \cap]$, which has both intersection and linked roles.

THEOREM 4.11. *Containment analysis in* $\mathsf{RT}[\twoheadleftarrow, \cap]$ *is in* **coNEXP**.

See Appendix A.5 for the proof. The proof shows that if there exists a counter example to the containment relation, i.e., there exists a reachable state in which the inclusion does not hold, then there exists such a reachable state of size at most exponential in the input size.

We have shown that containment analysis for $\mathsf{RT}[\twoheadleftarrow, \cap]$ is **PSPACE**-hard (from Theorem 4.4) and is in **coNEXP**. The exact complexity is still unknown. We have tried applying the approach for $\mathsf{RT}[\twoheadleftarrow]$ to the case of $\mathsf{RT}[\twoheadleftarrow, \cap]$, and we now discuss the difficulties of this approach. There is a natural mapping between $\mathsf{RT}[\twoheadleftarrow]$ and pushdown systems. $\mathsf{RT}[\twoheadleftarrow, \cap]$, which adds to $\mathsf{RT}[\twoheadleftarrow]$ intersection inclusion statements, can be mapped to alternating pushdown systems. An alternating pushdown system (APDS for short) is a triplet (Π, Γ, Δ), where Π and Γ are the same as for PDSs, and Δ is a function that assigns to each element of $(\Pi \times \Gamma)$ a negation-free boolean formula over elements of $\Pi \times \Gamma^*$. If $\Delta(q, \gamma) = (q_1, \omega_1) \wedge (q_2, \omega_2)$, then for every $\omega \in \Gamma^*$, the immediate successor of $\langle q, \gamma\omega\rangle$ is the set $\{\langle q_1, \omega_1\omega\rangle, \langle q_2, \omega_2\omega\rangle\}$. Intuitively, at the configuration $\langle q, \gamma\omega\rangle$ the APDS forks into two copies in the configurations $\langle q_1, \omega_1\omega\rangle$ and $\langle q_2, \omega_2\omega\rangle$. Because of the mapping from $\mathsf{RT}[\twoheadleftarrow, \cap]$ to APDSs, the $\mathsf{RT}[\twoheadleftarrow, \cap]$ containment analysis problem is reduced to determining containment of the reachable frontiers of two APDSs. Each frontier is a set

of configurations of a set of PDSs. It is known that given an APDS the set of all configurations that can reach a particular configuration can be encoded using an alternating finite automata (AFA). It is known that determining containment between two languages accepted by AFA's is **PSPACE**-complete. However, there exists no result on characterizing the frontiers reachable from a particular configuration. In particular, it is not clear how to encode all the reachable frontiers (a set of sets of strings) succinctly. [3]

5. DISCUSSIONS AND RELATED WORK

We have shown that containment analysis is intractable in $\mathsf{RT}[\cap]$, $\mathsf{RT}[\twoheadleftarrow]$, and $\mathsf{RT}[\twoheadleftarrow, \cap]$. 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 not difficult to extend our LP-based approach for containment analysis in $\mathsf{RT}[\,]$ to the case of $\mathsf{RT}[\twoheadleftarrow]$ and $\mathsf{RT}[\twoheadleftarrow, \cap]$, such that containment relationships in our running example 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 analysis can be solved efficiently. As discussed in Section 1, security analysis in the form of simple safety analysis has been studied in the HRU model [Harrison et al. 1976], 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 [Harrison et al. 1976], 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 [Harrison et al. 1976], 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 in the size of the access control matrix. However, if commands are a parameter to the problem, the safety problem is **NP**-complete.

[3] The difficulty of encoding such a set of frontiers was pointed out to us by Ahmed Bouajjani in personal communication.

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 simple member statements, then adding (or removing) $A.r \longleftarrow D$ corresponds to adding (or removing) r to the cell on row D and column A. Therefore, if we consider safety analysis in the sub-language of $\mathsf{RT}[\leftarrow, \cap]$ that has only simple member statements, this problem is a sub-problem of the HRU safety problem.

Adding a simple inclusion 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 linking inclusion 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 intersection inclusion 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, even when we restrict ourself to $\mathsf{RT}[\,]$, which has simple inclusion as well as simple member statements. 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 simple inclusion or an intersection inclusion statement, or one for every pair of objects when adding a linking inclusion 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 is 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 [Harrison et al. 1976].

> 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 state changes in which (parametric) restrictions are placed on allowed changes. In [Chander et al. 2001], 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 $\mathsf{RT}[\twoheadleftarrow, \cap]$ is closely related to SDSI, whose semantics and evaluation has been the subject of many previous works [Abadi 1998; Clarke et al. 2001; Halpern and van der Meyden 2001; Jha and Reps 2002; Li 2000; Li et al. 2003]. 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 [Halpern and van der Meyden 2001]. Both [Abadi 1998] and [Halpern and van der Meyden 2001] consider 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 [Jha and Reps 2002], evaluating queries given a set of SDSI statements is reduced to model checking pushdown systems; there, only a fixed set of SDSI statements is considered, which are encoded as transition rules in the automata. Other works [Clarke et al. 2001; Li 2000; Li et al. 2003] do not handle inclusion queries or consider restricted state changes.

The notion of delegation here is similar to the notion of "speaks for" in the ABLP logic for authentication and access control [Abadi et al. 1993; Lampson et al. 1992]. In ABLP logic, that A speaks for B means that, if principal A makes a statement, then we can believe that principal B makes it, too. That A speaks for B can be viewed as a delegation of all authority from B to A. The ABLP logic is designed mainly for authentication; its total delegation has too coarse a granularity for access control. To limit the authority being delegated in the logic, a principal can adopt a role before delegating. By using roles, one can achieve effects roughly similar to decentralized attributes and delegation of attribute

authority. However, SRC logic does not ave attribute-based delegation supported using linked roles.

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 analysis that capture both safety and availability.

Although the trust management primitives we consider are more expressive than some aspects of the HRU model [Harrison et al. 1976], our main results show that persistence of nontrivial safety and availability properties may be algorithmically tractable. Specifically, membership queries and boundedness 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 inclusion queries, we look at several cases involving different policy sub-languages. For $\mathsf{RT}[\,]$, 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 $\mathsf{RT}[\cap]$, which is $\mathsf{RT}[\,]$ plus intersection, the problem becomes $\mathbf{coNP}$-complete. Intuitively, the reason is that multiple statements about a role represent disjunction, while intersection of roles provides a corresponding form of conjunction. For $\mathsf{RT}[\twoheadleftarrow]$, which is $\mathsf{RT}[\,]$ plus role linking, role containment for all reachable policy states is $\mathbf{PSPACE}$-complete. For $\mathsf{RT}[\twoheadleftarrow, \cap]$, which includes role linking, the problem remains decidable; our current upper bound is $\mathbf{coNEXP}$ (or double-exponential time) and lower bound is $\mathbf{PSPACE}$-hard.

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 additional work remains. Exact complexity bound for containment analysis in $\mathsf{RT}[\twoheadleftarrow, \cap]$ is still open. 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.

Acknowledgement

This work is supported by DARPA through SPAWAR contracts N66001-00-C-8015 and N66001-01-C-8005. It is also supported by DOD MURI "Semantics Consistency in Information Exchange" as ONR Grant N00014-97-1-0505 and by DOD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795.

We thank Fred Schneider and Somesh Jha for discussions that led to our study of safety and availability properties in Trust Management. We also thank the anonymous reviewers for their encouragement and helpful comments.

REFERENCES

ABADI, M. 1998. On SDSI's linked local name spaces. *Journal of Computer Security 6*, 1–2, 3–21.

ABADI, M., BURROWS, M., LAMPSON, B., AND PLOTKIN, G. 1993. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems 15*, 4 (Oct.), 706–734.

APT, K. R., BLAIR, H. A., AND WALKER, A. 1988. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, Los Altos, CA, 89–148.

BLAZE, M., FEIGENBAUM, J., IOANNIDIS, J., AND KEROMYTIS, A. D. 1999a. The KeyNote trust-management system, version 2. IETF RFC 2704.

BLAZE, M., FEIGENBAUM, J., IOANNIDIS, J., AND KEROMYTIS, A. D. 1999b. The role of trust management in distributed systems. In *Secure Internet Programming*. Lecture Notes in Computer Science, vol. 1603. Springer, 185–210.

BLAZE, M., FEIGENBAUM, J., AND LACY, J. 1996. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 164–173.

BOUAJJANI, A., ESPARZA, J., AND MALER, O. 1997. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of CONCUR'97*. Number 1256 in Lecture Notes in Computer Science. Springer, 135–150.

CHANDER, A., DEAN, D., AND MITCHELL, J. C. 2001. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 27–43.

CLARKE, D., ELIEN, J.-E., ELLISON, C., FREDETTE, M., MORCOS, A., AND RIVEST, R. L. 2001. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security 9*, 4, 285–322.

DOWLING, W. F. AND GALLIER, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming 1*, 3, 267–284.

ELLISON, C., FRANTZ, B., LAMPSON, B., RIVEST, R., THOMAS, B., AND YLONEN, T. 1999. SPKI certificate theory. IETF RFC 2693.

ESPARZA, J., HANSEL, D., ROSSMANITH, P., AND SCHWOON, S. 2000. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*. LNCS, vol. 1855. Springer, 232–247.

GAREY, M. R. AND JOHNSON, D. J. 1979. *Computers And Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

GRAHAM, G. S. AND DENNING, P. J. 1972. Protection — principles and practice. In *Proceedings of the AFIPS Spring Joint Computer Conference*. Vol. 40. AFIPS Press, 417–429.

GUNTER, C. A. AND JIM, T. 2000. Policy-directed certificate retrieval. *Software: Practice & Experience 30*, 15 (Sept.), 1609–1640.

HALPERN, J. AND VAN DER MEYDEN, R. 2001. A logic for SDSI's linked local name spaces. *Journal of Computer Security 9*, 1-2, 47–74.

HARRISON, M. A., RUZZO, W. L., AND ULLMAN, J. D. 1976. Protection in operating systems. *Communications of the ACM 19*, 8 (Aug.), 461–471.

HUNT, H. B., ROSENKRANTZ, D. J., AND SZYMANSKI. 1976. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences 12*, 2 (Apr.), 222–268.

JHA, S. AND REPS, T. 2002. Analysis of SPKI/SDSI certificates using model checking. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 129–144.

JIM, T. 2001. SD3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 106–115.

LAMPSON, B., ABADI, M., BURROWS, M., AND WOBBER, E. 1992. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems 10*, 4 (Nov.), 265–310.

LAMPSON, B. W. 1971. Protection. In *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*. Reprinted in ACM Operating Systems Review, 8(1):18-24, Jan 1974.

LI, N. 2000. Local names in SPKI/SDSI. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2–15.

LI, N., GROSOF, B. N., AND FEIGENBAUM, J. 2003. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security 6*, 1 (Feb.), 128–171.

LI, N. AND MITCHELL, J. C. 2003a. Datalog with constraints: A foundation for trust management languages. In *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages (PADL 2003)*. Springer, 58–73.

LI, N. AND MITCHELL, J. C. 2003b. RT: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*. IEEE Computer Society Press.

LI, N., MITCHELL, J. C., AND WINSBOROUGH, W. H. 2002. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 114–130.

LI, N., WINSBOROUGH, W. H., AND MITCHELL, J. C. 2003. Distributed credential chain discovery in trust management. *Journal of Computer Security 11,* 1 (Feb.), 35–86.

LIPTON, R. J. AND SNYDER, L. 1977. A linear time algorithm for deciding subject security. *Journal of the ACM 24,* 3, 455–464.

LLOYD, J. W. 1987. *Foundations of Logic Programming, Second Edition.* Springer.

RIVEST, R. L. AND LAMPSON, B. 1996. SDSI — a simple distributed security infrastructure. Available at http://theory.lcs.mit.edu/∼rivest/sdsi11.html.

SANDHU, R. S. 1988. The schematic protection model: Its definition and analysis for acyclic attenuating systems. *Journal of the ACM 35,* 2, 404–432.

SANDHU, R. S. 1992. The typed access matrix model. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 122–136.

SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. 1996. Role-based access control models. *IEEE Computer 29,* 2 (February), 38–47.

WEEKS, S. 2001. Understanding trust management systems. In *Proceedings of 2001 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 94–105.

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

time.

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 an* RT[ ] *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 $(c1)$ 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 $(c1)$: $X.u \longleftarrow B.r_1 \in \mathcal{P}|_{\mathcal{R}}$ 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, because $\mathcal{P}'$ is reachable. As $SP(\mathcal{P}') \models m(A, r, E)$, 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 $Y.u = A.r$, as 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 $(c1)$, $fc(Y, u, ?Z, ?w) :- fc(Y_1, u_1, ?Z, ?w) \in BCP$. 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* RT[ ], $\mathcal{R}$, *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 $Z.w = A.r$, as 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 $SP(\mathcal{P}|_{\mathcal{R}}) \not\models m(Z_1, w_1, D)$, because 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 $(c1)$. □

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

LEMMA A.2. *Assume we are given an* RT[ ] *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. As $X.u$ 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 $(c1)$, 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 an* RT[ ] *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 $\mathcal{P}$ as $A.r$ 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 $m(B, r_1, E) \in T_{SP(\mathcal{P}')} \uparrow^i$. As $A.r$ 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 and 4.10

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 simple member 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 simple member 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 simple member statements. From $\mathcal{P}''$, we derive $\mathcal{P}'''$ by removing all simple member statements that use 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 exactly the same memberships as $\mathcal{P}''$. Intuitively, $A.v \longleftarrow D$ cannot affect members of roles in Roles($\mathcal{P}''$) unless the body of some statement refers to the role name v, which is impossible, as every statement in $\mathcal{P}'''$ that could have role names in its body is also in $\mathcal{P}$, and so does not use v. □

Theorem 4.3: *Containment analysis in* RT[∩] *is* **coNP***-complete.*

PROOF. To show **coNP**-hardness, we reduce the monotone 3SAT problem to the complement of the universal containment problem in RT[∩]. Monotone 3SAT is 3SAT with each clause containing either only positive literals or only negative literals; it is known to be **NP**-complete [Garey and Johnson 1979].

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 $1 \leq i \leq s$, $I(p_i) =$ 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 $\mathcal{P}' = \mathcal{P} \cup \{A.p_i \longleftarrow Z \mid 1 \leq i \leq s \wedge I(p_i) = \text{true}\}$. $\mathcal{P}'$ 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 $\mathsf{RT}[\cap]$ is in $\mathbf{coNP}$. 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 simple member 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 $\mathbf{coNP}$. The method we use to construct the counter example $\mathcal{P}''$ also yields an exponential algorithm for determining containment. □

Theorem 4.10:*Containment analysis in* $\mathsf{RT}[\twoheadleftarrow]$ *where all roles in* $\mathsf{Roles}(\mathcal{P})$ *are g-restricted is* $\mathbf{coNP}$*-complete.*

PROOF. As one can nondeterministically guess a subset $\mathcal{P}'$ of $\mathcal{P}$ and verify that the containment does not hold, the problem is clearly in $\mathbf{coNP}$. To prove $\mathbf{coNP}$-hardness, we reduce the monotone 3SAT problem to the complement of universal role containment in $\mathsf{RT}[\twoheadleftarrow]$; the reduction is similar to that in the proof of Theorem 4.3. 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 if 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. □

A.4 Proof of Proposition 4.6

Proposition 4.6: *Given a set $\mathcal{P}$ of* RT$[\leftarrow]$ *statements, $SP(\mathcal{P}) \models m(A, r, D)$ if and only if* RS$[\mathcal{P}] \models A\, r \overset{*}{\rightarrowtail} D$.

PROOF. We prove the only if part by using induction on i to show that if $m(A, r, D) \in T_{SP(\mathcal{P})}\uparrow^i$, then $RS(\mathcal{P}) \rhd A\, r \overset{*}{\rightarrowtail} D$. The basis is trivially satisfied because $T_{SP(\mathcal{P})}\uparrow^0 = \emptyset$. In the step, $m(A, r, D) \in T_{SP(\mathcal{P})}\uparrow^{i+1}$, one of $(m1)$, $(m2)$, $(m3)$ is used to derive this.

Case $(m1)$: $A.r \longleftarrow D \in SP(\mathcal{P})$, this means that $A\, r \mapsto D \in \mathcal{P}$. Clearly, RS$[\mathcal{P}] \rhd A\, r \overset{*}{\rightarrowtail} D$.

Case $(m2)$: $A.r \longleftarrow B.r_1 \in SP(\mathcal{P})$, and $m(B, r_1, D) \in T_{SP(\mathcal{P})}\uparrow^i$. In this case, $A\, r \mapsto B\, r_1 \in \mathcal{P}$, and by induction hypothesis, RS$[\mathcal{P}] \rhd B\, r_1 \overset{*}{\rightarrowtail} D$. Using rewriting rules in RS$[\mathcal{P}]$, one can rewrite $A\, r$ first to $B\, r_1$, and then to D; so RS$[\mathcal{P}] \rhd A\, r \overset{*}{\rightarrowtail} D$.

Case $(m3)$: $A.r \longleftarrow A.r_1.r_2 \in SP(\mathcal{P})$ and $m(A, r_1, E),\ m(E, r_2, D) \in T_{SP(\mathcal{P})}\uparrow^i$. By induction hypothesis, RS$[\mathcal{P}] \rhd B\, r_1 \overset{*}{\rightarrowtail} E,\ E\, r_2 \overset{*}{\rightarrowtail} D$. Using rewriting rules in RS$[\mathcal{P}]$, one can rewrite $A\, r$ first to $A\, r_1\, r_2$, then into $E\, r_2$, and finally into D.

We prove the if part by using induction on i to show that if using RS$[\mathcal{P}]$ one can rewrite $A\, r$ into D in i steps, then $SP(\mathcal{P}) \models m(A, r, D)$. Base case, $i = 1$ and $A.r \longleftarrow D \in \mathcal{P}$, clearly $SP(\mathcal{P}) \models m(A, r, D)$. Consider the step, consider the first rewriting step. One of the following two cases apply.

Case one: $credA.rB.r_1 \in \mathcal{P}$ is used in the first step. By induction hypothesis, $SP(\mathcal{P}) \models m(B, r_1, D)$. Furthermore, $m(A, r, ?X) :- m(B, r_1, ?X) \in SP(\mathcal{P})$; thus $SP(\mathcal{P}) \models m(A, r, D)$.

Case two: $credA.rA.r_1.r_2 \in \mathcal{P}$ is used in the first step. There must exist a principal E such that RS$[\mathcal{P}] \rhd A\ r_1\ r_2 \overset{*}{\rightarrowtail} E.r_2 \overset{*}{\rightarrowtail} D$. By induction hypothesis, $SP\mathcal{P} \models m(A, r_1, E),\ m(E, r_2, D)$. Furthermore, $m(A, r, ?X) :- m(A, r_1, ?Y), m(?Y, r_2, ?X) \in SP(\mathcal{P})$; thus $SP(\mathcal{P}) \models m(A, r, D)$.

□

A.5 Proof of Theorem 4.11

Theorem 4.11: *Containment analysis in* RT$[\leftarrow, \cap]$ *is in* **coNEXP**.

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 simple member 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 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 is 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 of 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 $O(M^2N)$ new simple member 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 **coNEXP**. 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 simple inclusion and linking inclusion 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 using the resulting set of principals 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 $\mathsf{Principals}(\mathcal{P}')$, 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, as $T_{SP(\mathcal{P}')}\uparrow^0 = \emptyset$. 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 statement $A.r \longleftarrow e$ that is not a simple member statement exists in $\mathcal{P}'$. It must also exist in $\mathcal{P}$, as $\mathcal{P}' - \mathcal{P}$ only has simple member statements; therefore, principals in $A.r \longleftarrow e$ are each in their own equivalence class. The statement must also exist in $\mathcal{P}''$, as 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 $SP(\mathcal{P}') \models m(X', u, E')$. As $X \in \mathsf{Principals}(\mathcal{P})$, 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, as $T_{SP(\mathcal{P}'')}\uparrow^0 = \emptyset$. 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 non-simple-member statement $A.r \longleftarrow e$ of $\mathcal{P}''$ is used; such a statement must be mapped from a non-simple-member statement in $\mathcal{P}'$. As 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, there exist Z_1' and Y_1' such that $SP(\mathcal{P}') \models m(Z_1', w_1, Y')$ 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 $[F'] = [F''] = F$. As $Z.w_1 \in \mathsf{SigRoles}(\mathcal{P}, \mathcal{P}', \mathcal{Q})$, 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 linking inclusion and intersection inclusion statements need to be collected. This may be used to explain why containment in $\mathsf{RT}[\,]$ is efficiently decidable.