



## Statements in $RT_0$

- Statements:
  - Type-1:  $A.r \leftarrow D$   
 $ABU.university \leftarrow StateU$
  - Type-2:  $A.r \leftarrow B.r_1$   
 $Store.university \leftarrow ABU.university$
  - Type-3:  $A.r \leftarrow A.r_1.r_2$   
 $Store.student \leftarrow Store.university.student$
  - Type-4:  $A.r \leftarrow B_1.r_1 \cap B_2.r_2$   
 $Store.discount \leftarrow Store.student \cap IEEE.member$

7

## The Abstract Analysis Problem

- Given an initial state  $P$ ,
  - a query  $Q$ ,
  - and a rule  $R$  that restricts how states can change (defines reachability among states);
- Is  $Q$  possible? (existential)
  - whether  $\exists$  reachable  $P'$  s.t.  $P' \triangleright Q$
- Is  $Q$  necessary? (universal)
  - whether  $\forall$  reachable  $P'$ ,  $P' \triangleright Q$

8

## Statements in $RT_0$ (1)

- Type-1:  $K.r \leftarrow K_1$ 
  - $mem[K.r] \supseteq \{K_1\}$
  - $K_{HR}.manager \leftarrow K_{Alice}$
- Type-2:  $K.r \leftarrow K_1.r_1$ 
  - $mem[K.r] \supseteq mem[K_1.r_1]$
  - $K_{SSO}.admin \leftarrow K_{HR}.manager$

9

## Statements in $RT_0$ (2)

- Type-3:  $K.r \leftarrow K_1.r_1.r_2$ 
  - Let  $mem[K.r_1]$  be  $\{K_1, K_2, \dots, K_n\}$   
 $mem[K.r] \supseteq mem[K_1.r_2] \cup mem[K_2.r_2]$   
 $\cup \dots \cup mem[K_n.r_2]$
  - $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
- Type-4:  $K.r \leftarrow K_1.r_1 \cap K_2.r_2$ 
  - $mem[K.r] \supseteq mem[K_1.r_2] \cap mem[K_2.r_2]$
  - $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$

10

## The Query $Q$

- Form-1:  $mem[K.r] \supseteq \{K_1, \dots, K_n\}$  ?
- Form-2:  $\{K_1, \dots, K_n\} \supseteq mem[K.r]$  ?
- Form-3:  $mem[K_1.r_1] \supseteq mem[K.r]$  ?

11

## The Restriction Rule $R$

- $R = (G, S)$ 
  - $G$  is a set of growth-restricted roles
    - if  $A.r \in G$ , then cannot add " $A.r \leftarrow \dots$ "
  - $S$  is a set of shrink-restricted roles
    - if  $A.r \in S$ , then cannot remove " $A.r \leftarrow \dots$ "
- Motivation:
  - Definitions of roles that are not under one's control may change

12

## Sample Analysis Queries

- Simple safety (existential form-1):
  - Is  $\text{mem}[K.r] \supseteq \{K_1\}$  possible?
- Simple availability (universal form-1):
  - Is  $\text{mem}[K.r] \supseteq \{K_1\}$  necessary?
- Bounded safety (universal form-2):
  - Is  $\{K_1, \dots, K_n\} \supseteq \text{mem}[K.r]$  necessary?
- Containment (universal form-3):
  - Is  $\text{mem}[K_1.r_1] \supseteq \text{mem}[K.r]$  necessary?

13

## How to Use Such Analysis

- Guarantee safety and availability properties of an AC system:
  - Properties one wants to guarantee are encoded in a set of queries
  - R represents how much control one has
    - parts not under one's control may change in R
    - parts under one's control are considered fixed in R
  - Before making changes, one can use analysis to guarantee properties are not violated

14

## Example

1.  $K_{SSO}.access \leftarrow K_{SSO}.admin$
2.  $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3.  $K_{SSO}.admin \leftarrow K_{HR}.manager$
4.  $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5.  $K_{HR}.employee \leftarrow K_{HR}.manager$
6.  $K_{HR}.employee \leftarrow K_{HR}.engineer$
7.  $K_{HR}.manager \leftarrow Alice$
8.  $Alice.access \leftarrow Bob$

Legend:     fixed  
              can grow, can shrink

15

## A Simple Availability Query

1.  $K_{SSO}.access \leftarrow K_{SSO}.admin$
2.  $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3.  $K_{SSO}.admin \leftarrow K_{HR}.manager$
4.  $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5.  $K_{HR}.employee \leftarrow K_{HR}.manager$
6.  $K_{HR}.employee \leftarrow K_{HR}.engineer$
7.  $K_{HR}.manager \leftarrow K_{Alice}$
8.  $Alice.access \leftarrow K_{Bob}$

Query:     Is  $\text{mem}[K_{SSO}.access] \supseteq \{K_{Alice}\}$  necessary?  
Answer:    Yes. (Available)  
Why:       Statements 1, 3, and 7 cannot be removed

## A Simple Safety Query

1.  $K_{SSO}.access \leftarrow K_{SSO}.admin$
2.  $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3.  $K_{SSO}.admin \leftarrow K_{HR}.manager$
4.  $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5.  $K_{HR}.employee \leftarrow K_{HR}.manager$
6.  $K_{HR}.manager \leftarrow K_{Alice}$
7.  $K_{HR}.employee \leftarrow K_{HR}.engineer$
8.  $Alice.access \leftarrow K_{Bob}$

Query:     Is  $\text{mem}[K_{SSO}.access] \supseteq \{K_{Eve}\}$  possible?  
Answer:    Yes. (Unsafe)  
Why:       Both  $K_{HR}.engineer$  and  $Alice.access$  may grow.

## A Containment Analysis Query about Safety

1.  $K_{SSO}.access \leftarrow K_{SSO}.admin$
2.  $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3.  $K_{SSO}.admin \leftarrow K_{HR}.manager$
4.  $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5.  $K_{HR}.employee \leftarrow K_{HR}.manager$
6.  $K_{HR}.employee \leftarrow K_{HR}.engineer$
7.  $K_{HR}.manager \leftarrow K_{Alice}$
8.  $Alice.access \leftarrow K_{Bob}$

Query:     Is  $\text{mem}[K_{HR}.employee] \supseteq \text{mem}[K_{SSO}.access]$  necessary?  
Answer:    Yes. (Safe)  
Why:        $K_{SSO}.access$  and  $K_{SSO}.admin$  cannot grow and Statement 5 cannot be removed.

## An Containment Analysis Query about Availability

1.  $K_{SSO}.access \leftarrow K_{SSO}.admin$
2.  $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3.  $K_{SSO}.admin \leftarrow K_{HR}.manager$
4.  $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5.  $K_{HR}.employee \leftarrow K_{HR}.manager$
6.  $K_{HR}.employee \leftarrow K_{HR}.engineer$
7.  $K_{HR}.manager \leftarrow K_{Alice}$
8.  $Alice.access \leftarrow K_{Bob}$

Query: Is  $mem[K_{SSO}.access] \supseteq mem[K_{HR}.manager]$  necessary?

Answer: Yes. (Available)

Why: Statements 1 and 3 cannot be removed

19

## Form-1 and Form-2 Queries

### PTIME

- Form-1 queries are monotonic in P
- Form-2 queries are anti-monotonic in P
- Use the minimal reachable state to answer universal form-1 and existential form-2
- The maximal reachable state answers existential form-1 and universal form-2
  - the state is simulated by a logic program

Reminder: Form-1 query:  $mem[K.r] \supseteq \{K_1, \dots, K_n\}$   
 Form-2 query:  $\{K_1, \dots, K_n\} \supseteq mem[K.r]$

20

## Universal Form-3 $\equiv$ Containment Analysis

- With just type 1 and 2 statements
  - containment analysis is in PTIME
    - using logic programs with stratified negation
- With type 1, 2, and 4 statements
  - containment analysis is coNP-complete
    - equivalent to determining validity of propositional-logic formulas

Reminder:

Queries: Form-3:  $mem[K_1.r_1] \supseteq mem[K.r]$

Statements: Type-1:  $K.r \leftarrow K_1$

Type-2:  $K.r \leftarrow K_1.r_1$

Type-4:  $K.r \leftarrow K_1.r_1 \cap K_2.r_2$

## Universal Form-3 (Containment Analysis)

- LRT (type 1, 2, and 3 statements)
  - containment analysis is PSPACE-complete
    - LRT  $\leftrightarrow$  string-rewriting systems
    - equivalent to determining containment of languages accepted by NFA's
  - remains PSPACE-complete without shrinking
  - coNP-complete without growing

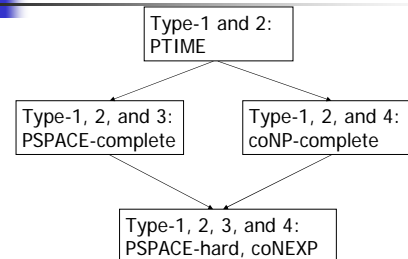
Reminder: Type-1:  $K.r \leftarrow K_1$   
 Type-2:  $K.r \leftarrow K_1.r_1$   
 Type-3:  $K.r \leftarrow K_1.r_1 \cap K_2.r_2$

## Universal Form-3 (Containment Analysis)

- SRT (all four types of statements)
  - in coNEXP
    - although infinitely many new principals and statements may be added, if the containment does not hold, there exists a counter example whose size is at most exponential
  - PSPACE-hard
  - exact complexity still open!
  - coNP-complete without growing

23

## Complexity Summary



24

## Mapping the HRU model to the Abstract Analysis Problem

- P: an access matrix
- R: the protection system state can change by executing commands
  - e.g.,  $c(x,y,z) \{ \text{if 'own'} \in \text{cell}(x,z) \wedge \text{'controls'} \in \text{cell}(x,y) \text{ then add 'read' to cell}(y,z) \}$
- Q: is  $r \in \text{cell}(s,o)$  possible?
  - simple safety queries only
- Main result in the HRU model
  - simple safety is undecidable

25

## Can HRU simulate SRT?

- Role memberships determined by a SRT state P is an access matrix
  - principals correspond to both subjects and objects
  - $K_1 \in \text{mem}[K.r]$ 
    - ⇔ subject  $K_1$  has right  $r$  over object  $K$
    - ⇔  $r \in \text{cell}(K_1, K)$
- Adding a type-1 statement  $K.r \leftarrow K_1$ 
  - adding  $r$  into  $\text{cell}(K_1, K)$

26

## Relating SRT with HRU

- Adding a type-2 statement  $K.r \leftarrow K_1.r_1$ 
  - for every  $K'$  such that  $K' \in \text{mem}[K_1.r_1]$  add  $r$  into  $\text{cell}(K', K)$
  - need to run an HRU command for every principal
  - this propagation needs to happen every time the matrix is changed

27

## Can HRU simulate SRT? (Probably not!)

- It seems that HRU cannot simulate SRT
  - Adding one statement corresponds to executing multiple HRU commands
  - Seems unable to simulate the effect of propagation
  - Unclear how to simulate removal of statements

28

## Why Our Problem is Decidable?

- Note that we consider queries that are more complicated than simple safety
  - e.g., containment analysis
- Some parameters in our analysis problem are simpler
  - no need to consider arbitrary commands
    - only four types of statements
  - restriction rules are static

29

## $RT_1 = RT_0 + \text{Parameterized Roles}$

- Motivations: to represent
  - role templates, e.g., course instructors, project leaders
  - relationships between entities, e.g., manages
  - roles and attributes that have fields, e.g., digital ids, diplomas

30

## RT<sub>1</sub>

- Approach:
  - a role name  $R$  takes the form  $r(h_1, \dots, h_n)$ , in which
    - $r$  is a role identifier and each of  $h_1, \dots, h_n$  is a data term (a constant or a variable) of the appropriate data type
  - data types include integer types, float types, enumeration types, and data/time types

31

## RT<sub>1</sub> (Examples)

- Example 1: Alpha allows manager of an employee to evaluate the employee
 

```
Alpha.evaluatorOf(?y) ← Alpha.managerOf(?y)
```
- Example 2: StateU gives special privileges to alumni who graduated during certain years:
 

```
StateU.foundingAlumni ←  
StateU.diploma(?degree, ?Year ∈ [1958..1962])
```

32

## RT<sub>2</sub> = RT<sub>1</sub> + Logical Objects

- Motivations:
  - to group logically related objects together and assign permissions about them together
- Approach: introducing o-sets, which are
  - similar to roles, but have values that are sets of things other than entities
  - defined through o-set definition credentials

33

## RT<sub>2</sub> (Examples)

- Example 1: Alpha allows members of a project team to read documents of this project
 

```
Alpha.documents(projectB) ←  
  "design_Doc_for_projectB"  
Alpha.team(projectB) ← Bob  
Alpha.fileAccess(read, ?F ∈  
  Alpha.documents(?proj)) ←  
  Alpha.team(?proj)
```

34

## RT<sub>2</sub> (Examples)

- Example 2: Alpha allows manager of the owner of a file to access the file
 

```
Alpha.read(?F) ←  
  Alpha.manager(?E ∈ Alpha.owner(?F))
```

35

## RT<sup>T</sup> Motivation: Separation of Duties (SoD)

- SoD requires **two or more different** persons be responsible for the completion of a sensitive task
- Previous TM systems use threshold structures for SoD
  - $k$  out of one set of entities
  - cannot express: require a manager and an accountant

36

## RT<sup>T</sup> Motivation

- RBAC uses mutually exclusive roles
  - no entity is allowed to occupy two mutually exclusive roles
  - a sensitive task requires a set of mutually exclusive roles
  - cannot express: require 3 managers
  - is non-monotonic

37

## RT<sup>T</sup> (The Approach)

- Introduce manifold roles
  - each member of a manifold role is a set of entities
- Introducing two new types of credentials:
  5.  $A.R \leftarrow B_1.R_1 \odot \dots \odot B_k.R_k$   
Multiple-role concurrence
  6.  $A.R \leftarrow B_1.R_1 \otimes \dots \otimes B_k.R_k$   
Separation of duties

38

## RT<sup>T</sup> (Examples)

- Example 1: require a manager and an accountant
  - $A.\text{approval} \leftarrow A.\text{manager} \odot A.\text{accountant}$
  - $\text{members}(A.\text{approval}) \supseteq \{\{x,y\} \mid x \in A.\text{manager}, y \in A.\text{accountant}\}$

39

## RT<sup>T</sup> (Examples)

- Example 2: require a manager and a *different* accountant
  - $A.\text{approval} \leftarrow A.\text{manager} \otimes A.\text{accountant}$
  - $\text{members}(A.\text{approval}) \supseteq \{\{x,y\} \mid x \neq y, x \in A.\text{manager}, y \in A.\text{accountant}\}$

40

## RT<sup>T</sup> (Examples)

- Example 3: require three different managers
  - $A.\text{approval} \leftarrow A.\text{manager} \otimes A.\text{manager} \otimes A.\text{manager}$
  - $\text{members}(A.\text{approval}) \supseteq \{\{x,y,z\} \mid x \neq y \neq z \in A.\text{manager}\}$

41

## RT<sup>D</sup>: Selective Use of Role Memberships

- Motivation: Selective use of role memberships (capacities) when making a request, and delegation of these capacities
- RBAC has session and role activation (user-to-session delegation)
  - but not process-to-process delegation

42

## RT<sup>D</sup>: The Approach

- Approach in RT<sup>D</sup>: introducing dynamic delegation credentials

$\gamma: B_1 \rightarrow B_2: D \text{ as } A.R$

43

## An Example

$S.delete(fileA) \leftarrow S.user \otimes S.goodWorkStation$

$S.user \leftarrow K\_Alice$

$S.goodWorkStation \leftarrow K\_ws$

$K\_Alice \rightarrow K\_p: K\_Alice \text{ as } S.user$

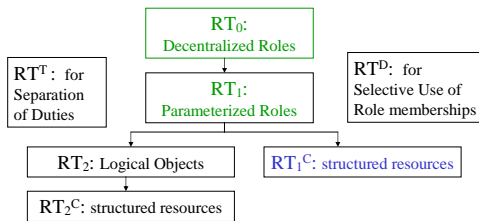
$K\_ws \rightarrow K\_p: K\_ws \text{ as } S.goodWorkStation$

$K\_p \rightarrow K\_ch: K\_ws \text{ as } S.goodWorkStation, K\_Alice \text{ as } S.user$

$K\_ch \rightarrow delete(fileA): K\_ws \text{ as } S.goodWorkStation, K\_Alice \text{ as } S.user$

44

## The Languages in the RT Framework



RT<sup>T</sup> and RT<sup>D</sup> can be used (either together or separately) with any of the five base languages: RT<sub>0</sub>, RT<sub>1</sub>, RT<sub>2</sub>, RT<sub>1</sub><sup>C</sup>, and RT<sub>2</sub><sup>C</sup>