

# Specification and Classification of Role-based Authorization Policies

**Gail-Joon Ahn**

**Laboratory of Information Integration, Security and  
Privacy (LIISP)**

**Univ. of North Carolina at Charlotte**



Center of Academic Excellence in Information Assurance Education



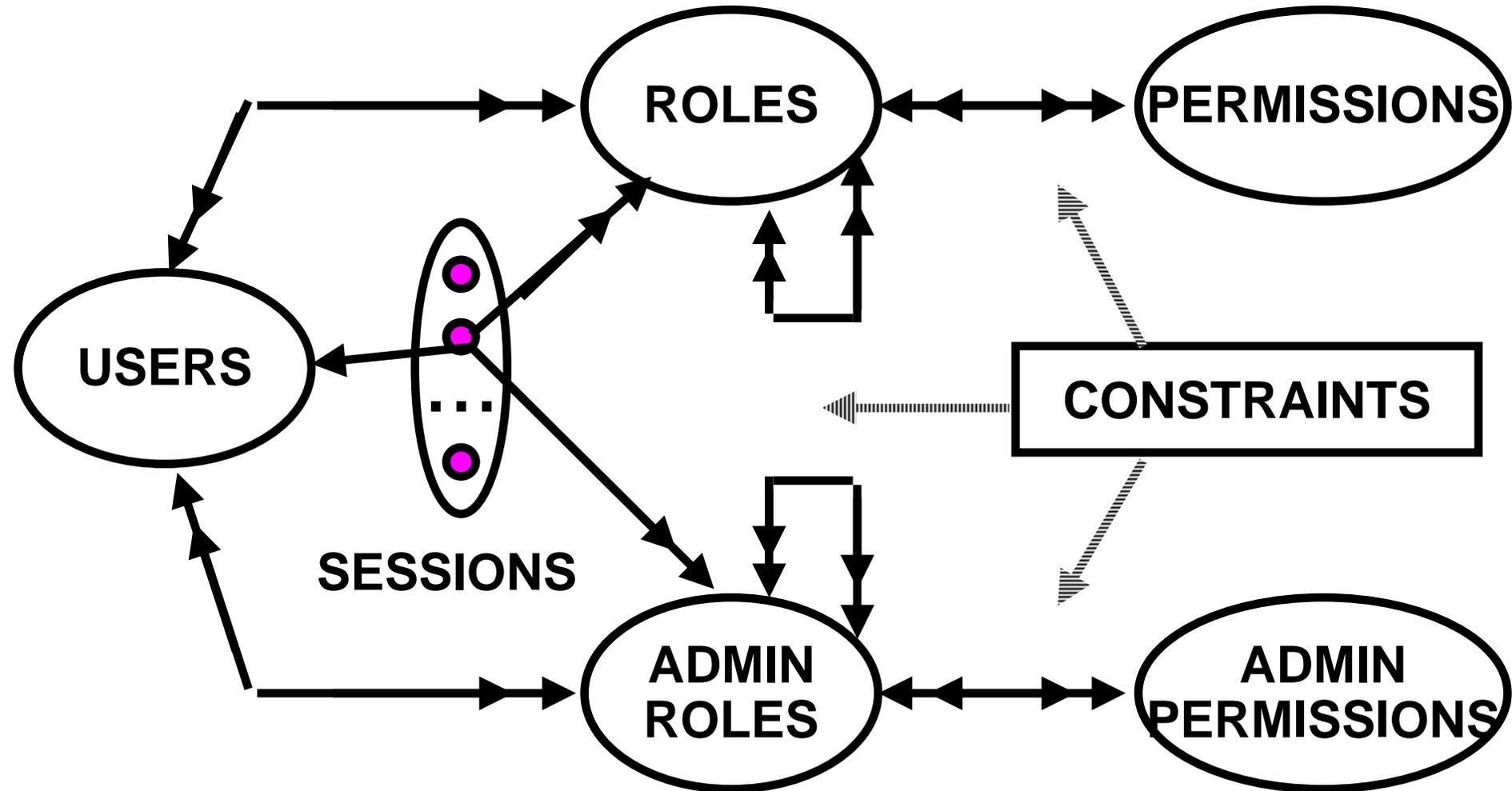


# OUTLINE

- ◆ Introduction
- ◆ Our approach
  - RCL2000
- ◆ Constraints specification
- ◆ Constraints classification
- ◆ Summary and ongoing research

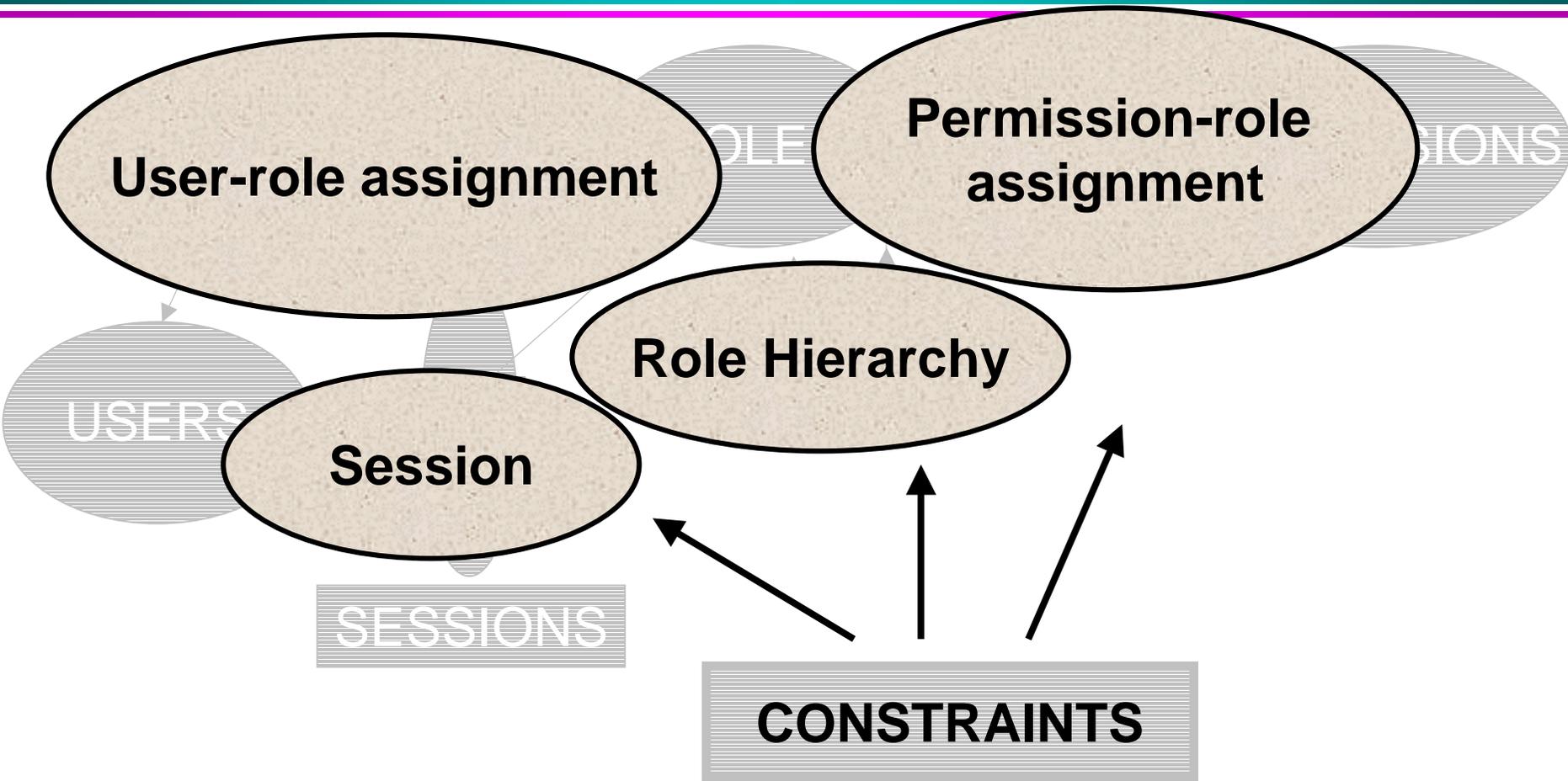


# RBAC96





# RBAC96





# SEPARATION OF DUTY (1)

- ◆ **SOD is fundamental technique for preventing fraud and errors**
  
- ◆ **Related Work**
  - **Enumerate several forms of SOD**
  - **Little work on specifying SOD in a comprehensive way**



# SEPARATION OF DUTY (2)



**PURCHASING  
MANAGER**



**ACCOUNTING PAYABLE  
MANAGER**



# OUR APPROACH

- ◆ **Need to specify these constraints**
  - **Language**
- ◆ **Show the meaning of expression**
  - **Formal semantics**
- ◆ **Expressive power of the language**
  - **Well-known constraints and simulations**
- ◆ **Analysis of the work**
  - **Characterization**



# BIG PICTURE

**Constraint Specification**

**Constraint Analysis**

**Constraint Enforcement**



# RCL 2000

- ◆ **RCL 2000 (Role-based Constraints Language 2000)**
- ◆ **Specification Language**
  - to formally express constraints in role-based systems
- ◆ **Most components are built upon RBAC96**
  - ACM Transactions on Information and System Security, Volume 3, No. 4, ACM, November 2000.

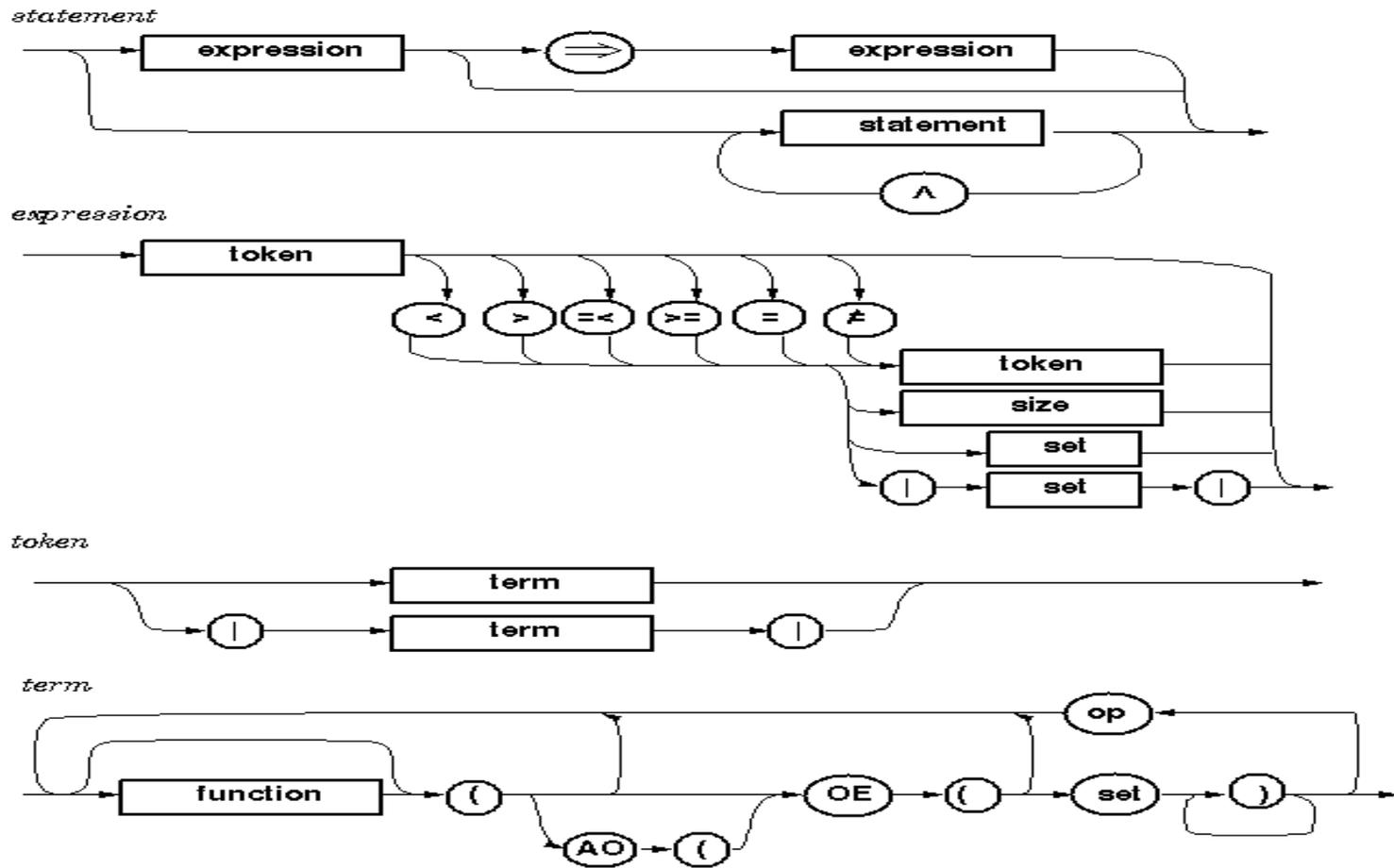


# NON-DETERMINISTIC FUNCTIONS

- ◆ introduced by Chen and Sandhu (1995)
- ◆ oneelement (OE)
  - $\text{oneelement}(X) = x_i$ , where  $x_i \in X$
- ◆ allother (AO)
  - $\text{allother}(X) = X - \{\text{OE}(X)\}$   
 $= X - \{x_i\}$
  - should occur along with OE function



# SYNTAX





# FORMAL SEMANTICS

## ◆ Reduction Algorithm

- to convert a constraint expression to a restricted form of first order predicate logic (RFOPL)

## ◆ Construction Algorithm

- to construct a constraint expression from RFOPL



# SOUNDNESS AND COMPLETENESS

- ◆ **Theorem 1** *Given RCL2000 expression  $\alpha$ ,  $\alpha$  can be translated into RFOPL expression  $\beta$ . Also  $\alpha$  can be reconstructed from  $\beta$ .*

$$\mathbf{C(R(\alpha)) = \alpha}$$

- ◆ **Theorem 2** *Given RFOPL expression  $\beta$ ,  $\beta$  can be translated into RCL2000 expression  $\alpha$ . Also  $\beta'$  which is logically equivalent to  $\beta$  can be reconstructed from  $\alpha$ .*

$$\mathbf{R(C(\beta)) = \beta'}$$



# SEPARATION OF DUTY CONSTRAINTS

- ◆ **Specification of SOD constraints identified by Simon and Zurko (1997) and formulated by Virgil et al (1998)**
- ◆ **Identify new SOD properties**
  - **Role-centric**
  - **User-centric**
  - **Permission-centric**



# ROLE-CENTRIC SOD CONSTRAINT EXPRESSION

## ◆ Static SOD

: Conflicting roles cannot have common users

$$U = \{u_1, u_2, \dots, u_n\}, \quad R = \{r_1, r_2, \dots, r_n\},$$

$$CR = \{cr_1, cr_2\} : cr_1 = \{r_1, r_2, r_3\}, \quad cr_2 = \{r_a, r_b, r_c\}$$

- $|\text{roles}(\text{OE}(U)) \cap \text{OE}(CR)| \leq 1$



# PERMISSION-CENTRIC SOD CONSTRAINT EXPRESSION

## ◆ SSOD-CP

- $|\text{permissions}(\text{roles}(\text{OE}(\text{U}))) \cap \text{OE}(\text{CP})| \leq 1$

## ◆ Variations of SSOD-CP

- SSOD-CP  $\wedge$

$$|\text{permissions}(\text{OE}(\text{R})) \cap \text{OE}(\text{CP})| \leq 1$$



# USER-CENTRIC SOD CONSTRAINT EXPRESSION

## ◆ SSOD-CU (User-centric)

- $SSOD-CR \wedge |\text{user}(OE(CR)) \cap OE(CU)| \leq 1$



# CASE STUDIES

- ◆ **Lattice-based access control**
  - Ravi Sandhu (1993, 1996)
- ◆ **Chinese Wall policy**
  - Ravi Sandhu (1992)
- ◆ **Discretionary access control**
  - Sandhu and Munawer (1998)



# LATTICE-BASED ACCESS CONTROL

H  
|  
L

HR  
|  
  
|  
LR

LW  
|  
  
|  
HW

- ◆ Subject  $s$  can write object  $o$  only if  $\lambda(s) \leq \lambda(o)$
- ◆ Subject  $s$  can read object  $o$  only if  $\lambda(o) \leq \lambda(s)$

**Constraints on UA:** *Each user is assigned to exactly two roles  $xR$  and  $LW$*



# LATTICE-BASED ACCESS CONTROL

- $AR = \{ar1, ar2\}$ 
  - $ar1 = \{HR, HW\}$ ,  $ar2 = \{LR, LW\}$
- $ASR = \{asr1, asr2\}$ 
  - $asr1 = \{HR, LW\}$ ,  $asr2 = \{LR, LW\}$

## ◆ Constraint on UA:

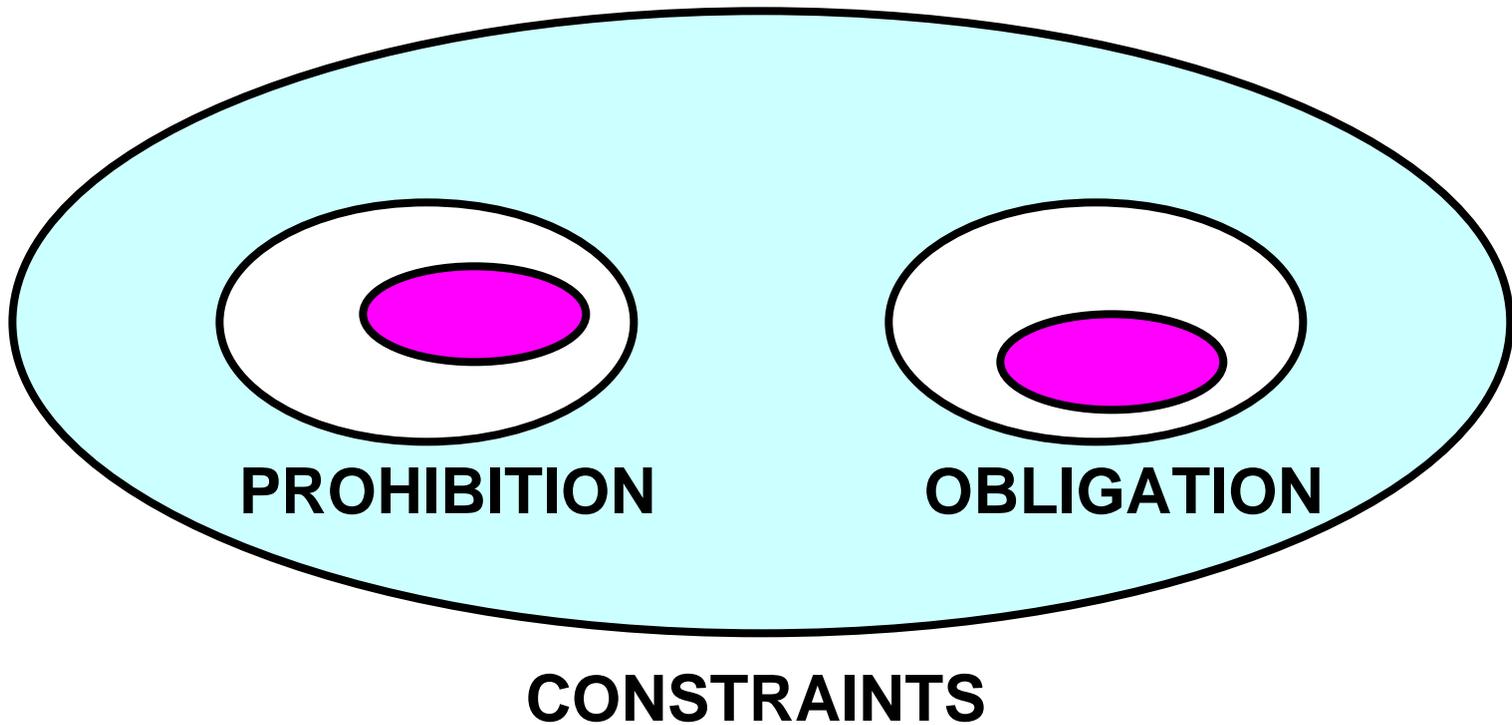
- $roles(OE(U)) = OE(ASR)$

## ◆ Constraint on sessions:

- $roles(OE(sessions(OE(U)))) = OE(AR)$



# CONSTRAINTS CLASSIFICATION





# PROHIBITION CONSTRAINTS

- ◆ **Forbid the RBAC component from doing (or being) something which is not allowed to do (or be)**
  - **Separation of duty constraints**



# OBLIGATION CONSTRAINTS

- ◆ **Force the RBAC component to do (or be) something**
  - **LBAC-RBAC, Chinese Wall-RBAC simulation**



# SIMPLE PROHIBITION CONSTRAINTS

## ◆ Type 1

- $|expr| \leq 1$

## ◆ Type 2

- $expr = \phi$  or  $|expr| = 0$

## ◆ Type 3

- $|expr1| < |expr2|$



# SIMPLE OBLIGATION CONSTRAINTS

## ◆ Type 1

- $expr \neq 0$  or  $|expr| > 0$

## ◆ Type 2

- Set X = Set Y

## ◆ Type 3

- obligation constraints  $\Rightarrow$  obligation constraints

## ◆ Type 4

- $|expr| = 1$ 
  - $|expr| = 1 \equiv |expr| \leq 1 \wedge |expr| > 0$



# SUMMARY

- ◆ **Developed the formal and intuitive language for role-based authorization constraints**
- ◆ **Provided a formal semantics for this language**
- ◆ **Demonstrated the expressive power of the language by**
  - specifying well-known separation of duty constraints
  - identifying new role-based SOD constraints
  - showing how to specify constraints identified in the simulations of other policies in RBAC
- ◆ **Characterized role-based constraints into prohibition and obligation constraints**



# ONGOING RESEARCH

- ◆ **Security policy management**
  - **Specification tool**
    - **PoMaTo**
  - **Architecture**
    - **Hive**



# PoMaTo

## Security Policy Management

The screenshot displays the PoMaTo: Policy Management Tool interface, which is divided into two main windows.

The top window, titled "PoMaTo: Policy Management Tool", shows the "Constraint Constructor" tab. It features a rule definition: "if a User is a member of Conflicting User then the User can be assigned to". A dropdown menu is open, listing options: "another Conflicting Role in the same Conflicting Role Set", "another Conflicting Permission in the same Conflicting Permission Set", "another Conflicting User in the same Conflicting User Set", and "another Role with Conflicting Permissions". The "OK" button is highlighted.

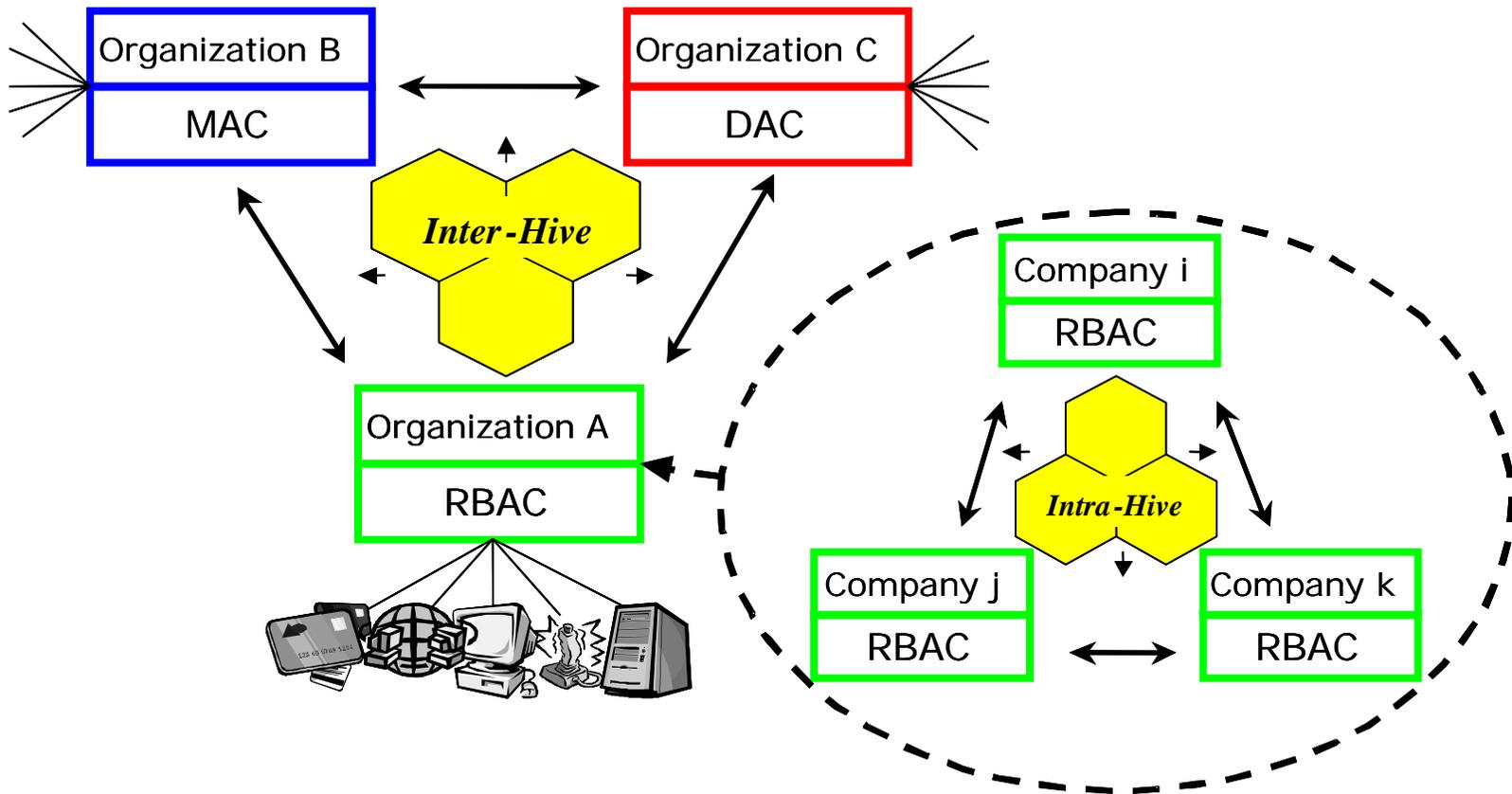
The bottom window, also titled "PoMaTo: Policy Management Tool", shows the "RBAC Environment" tab. It contains three sections for managing the environment:

- Enter the Roles and assign them to Conflicting Role(CR) set:** A list of roles includes "Admin" and "Top Officer". Buttons for "Add", "Delete selected", and "Assign to CR >>" are present.
- Enter the Permissions and assign them to Conflicting Permission(CP) set:** A list of permissions is shown. Buttons for "Add", "Delete Selected", and "Assign to CP >>" are present.
- Enter the Users and assign them to Conflicting User(CU) set:** A list of users is shown. Buttons for "Add", "Display", "Delete selected", and "Assign to CU >>" are present.



# Hive

## Security Policy Management





# FUTURE WORK

- ◆ **Extension of RCL 2000**
  - Applying it the formalization of some enterprise security policies
- ◆ **Implementation Issue**
  - Tool for checking syntax and semantic as well as visualization of specification
- ◆ **Enforcement of constraints**
- ◆ **Integration with meta-level assertion languages**