Access Control - II

CSE 565: Fall 2024

Computer Security

Xiangyu Guo (xiangyug@buffalo.edu)

University at Buffalo

Acknowledgement

- We don’t claim any originality of this slides. The content is developed heavily based on
    - Slides from Prof Ziming Zhao’s past offering of CSE565 (https://zzm7000.github.io/teaching/2023springcse410565/index.html)
    - Slides from Prof Marina Blanton’s past offering of CSE565 (https://www.acsu.buffalo.edu/~mblanton/cse565/)
    - Slides from Prof Hongxin Hu’s past offering of CSE565

Review of Last Lecture

- Access control principles
  - Access control matrices
  - Access control lists (ACLs): object-oriented.
  - Capability tickets: user-oriented.
- POSIX File Permissions
- Discretionary access control
  - Let's subjects to grant privileges to other subjects at their discretion

Today's Topic

- Mandatory access control (MAC)
- Role-based access control (RBAC)
- Attribute-based access control (ABAC)

Mandatory Access Control (MAC)

Recall: Discretionary Access Control

- **Owner-based Control**
  - Each object (e.g., files, directories) has an owner, typically the creator of the object.
- **Flexible Delegation**
  - The owner can delegate access to other users. The granted permission can further propagate.
- **Identity-based Access**
  - Access control is typically based on user identity or group membership.

Mandatory Access Control

- In Mandatory Access Control (MAC) users are granted privileges, which they ***cannot*** *control or change*
  - Useful for military applications
  - Useful for regular operating systems
- DAC does not protect against
  - Malware, Software bugs, or Malicious local users
- The SELinux enhancement to the Linux kernel implements the Mandator Access Control (MAC) policy, which allows you to define a security policy that provides granular permissions for all users, programs, processes, files, and devices

MAC in Operating Systems

- The need for MAC
  - Host compromised by network-based attacks is the root cause of many serious security problems
    - worm, botnet, DDoS, phishing, spamming
- Hosts can be easily compromised
  - Programs contain exploitable bugs
  - DAC mechanisms in OSs were not designed to take buggy software in mind
- Adding MAC to OSs is essential to deal with host compromise
  - Last line of defense when everything else fails
  - In MAC a system-wide security policy restricts access rights of subjects

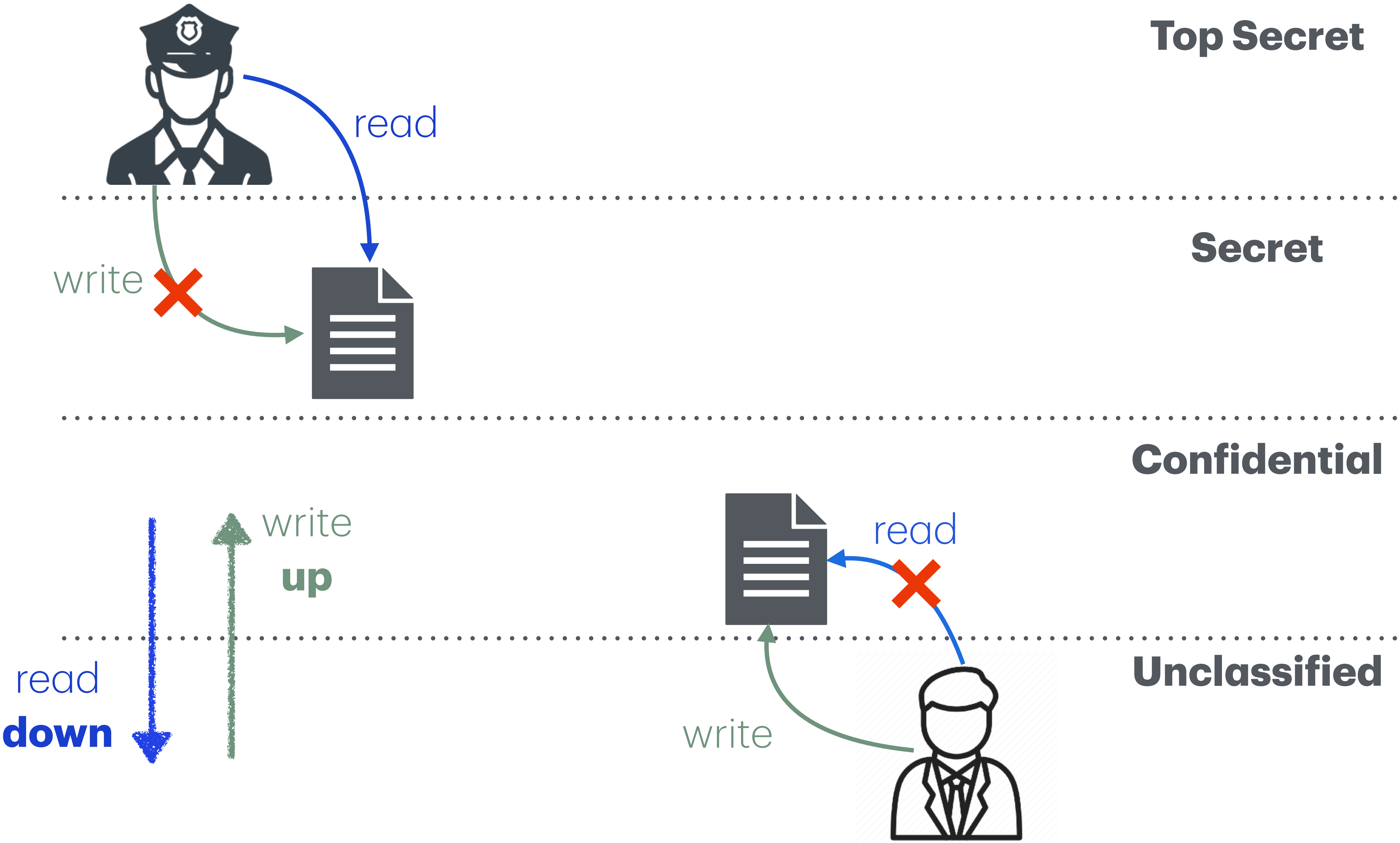
Combining MAC and DAC

- It is common to combine mandatory and discretionary access control in complex systems
- Modern operating systems is one significant example
- MAC and DAC are also combined in older models that implement multilevel security (for military-style security classes)
  - Bell-Lapadula confidentiality model (1973)
  - Biba integrity model (1977)

Bell-LaPadula (BLP) Confidentiality Model

- Security Level
  - No Read Up
  - No Write Down
- Security Category
  - Only access data *within* designated category
- Discretionary Access Control: Fine-tuned control over access to objects.

BLP Model: Security Levels



Top Secret

Secret

Confidential

Unclassified

read

write

write

up

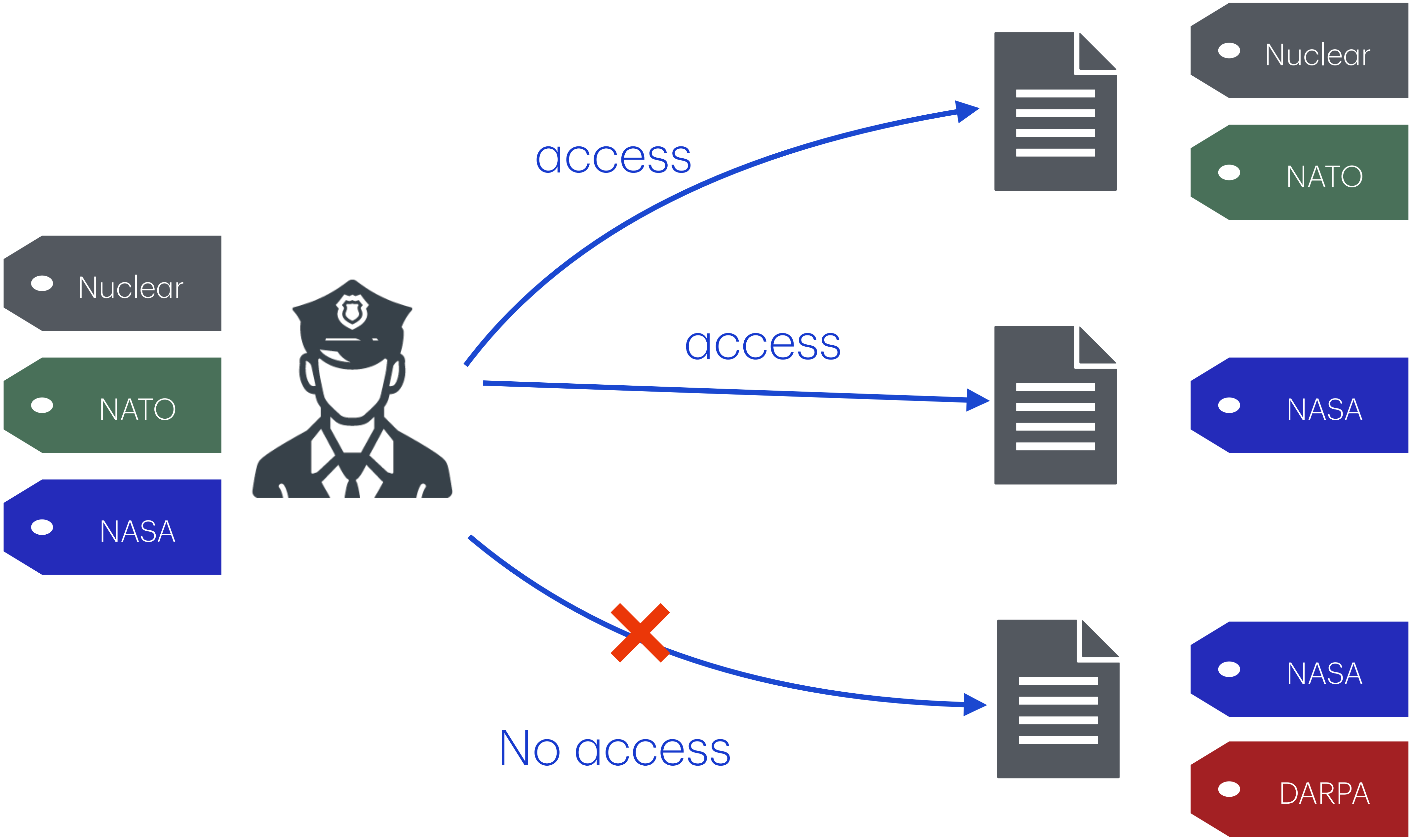
read

down

read

write

BLP Model: Security Categories



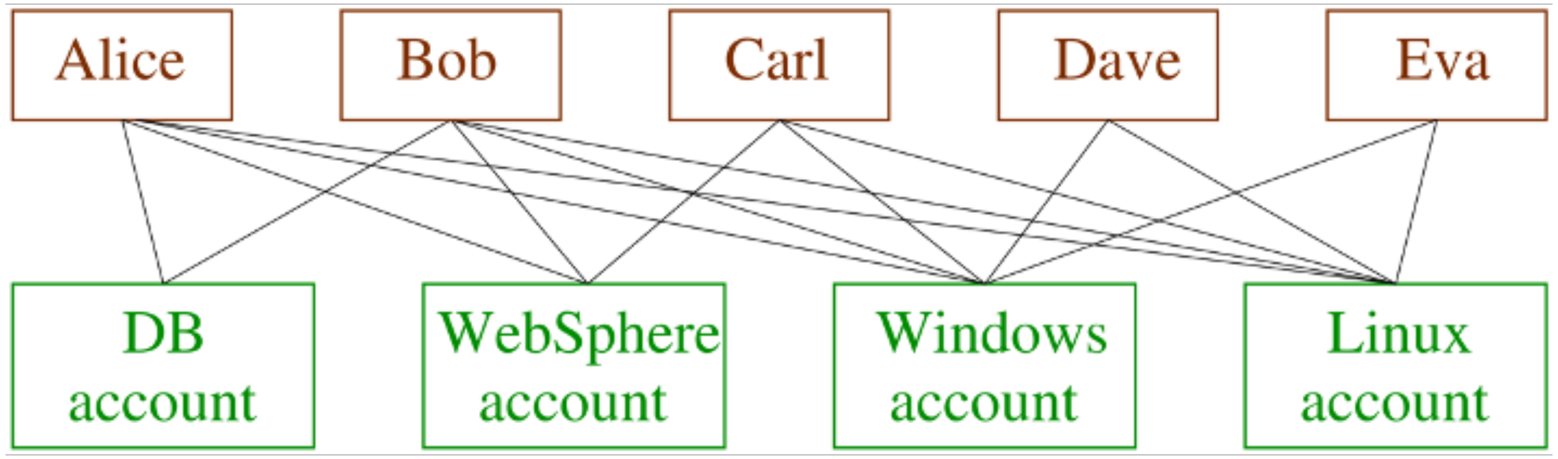
Role-Based Access Control (RBAC)

Role-Based Access Control

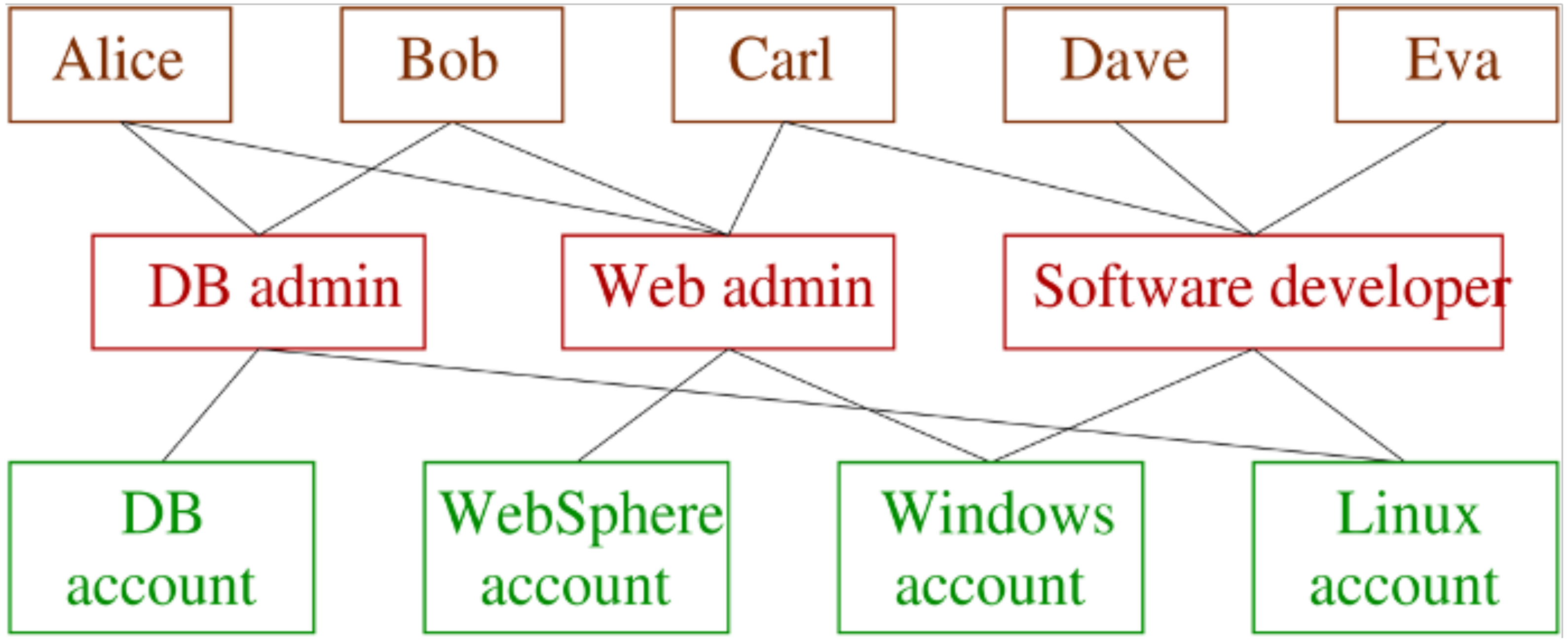
- In Role-Based Access Control (RBAC) models, subjects are combined into “roles” according to their privileges in the organization
    - Often based on job function
    - Permissions are assigned to roles rather than users
    - A user can assume one or more roles within the organization according to their responsibilities
- RBAC fits operational model of an organization and is widely used

Role-Based Access Control

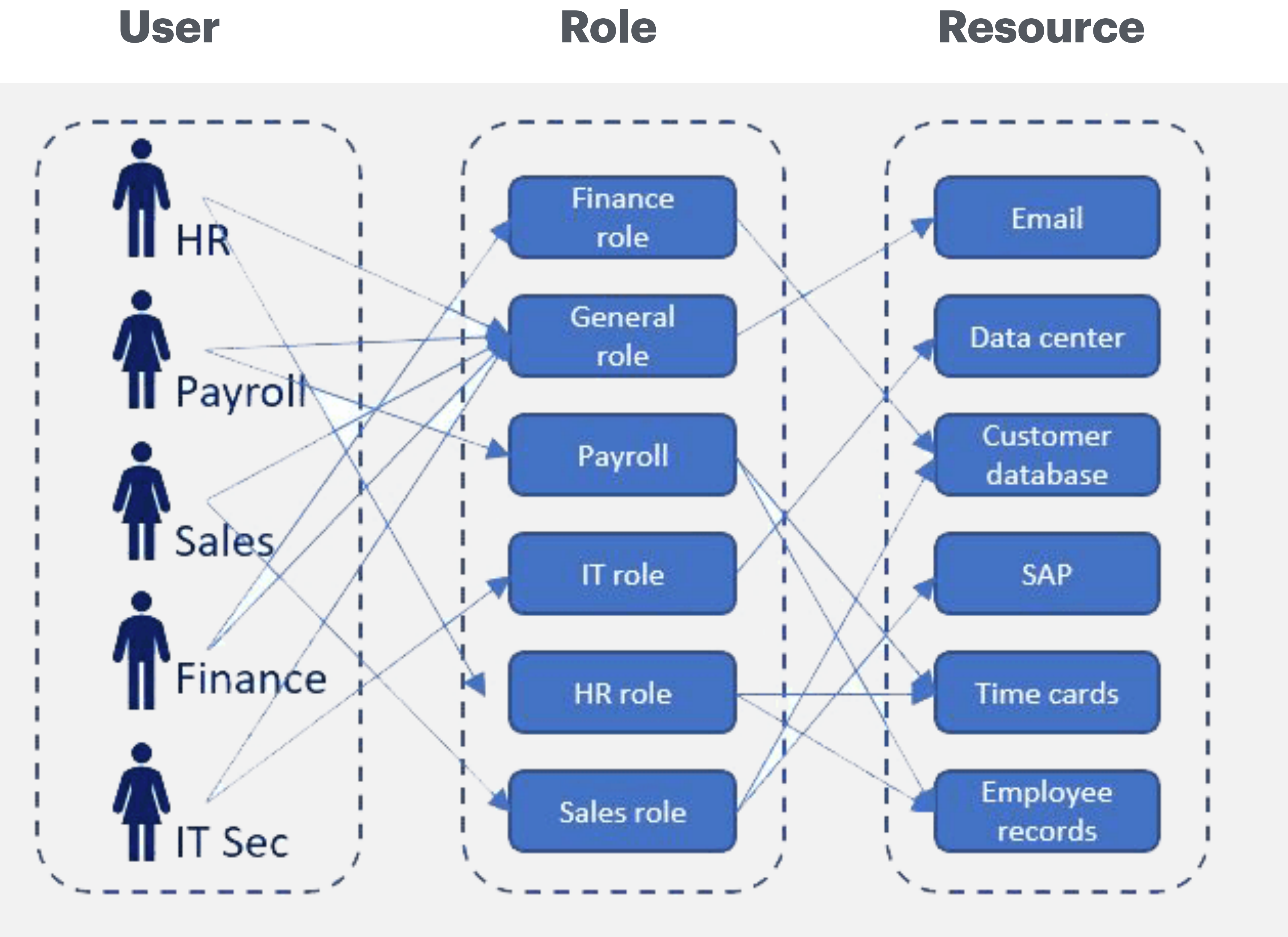
Non-role-based AC:



Role-based AC:



Role-Based Access Control



Role-Based Access Control

- Motivation for RBAC
  - Problem: it is difficult to manage user–permission relation
  - Roles are a level of indirection
    - *"All problems in Computer Science can be solved by another level of indirection"* - B. Lampson
- RBAC is
  - Multi-faceted & Multi-dimensional
  - Open ended
  - Ranging from simple to sophisticated

Role-Based Access Control

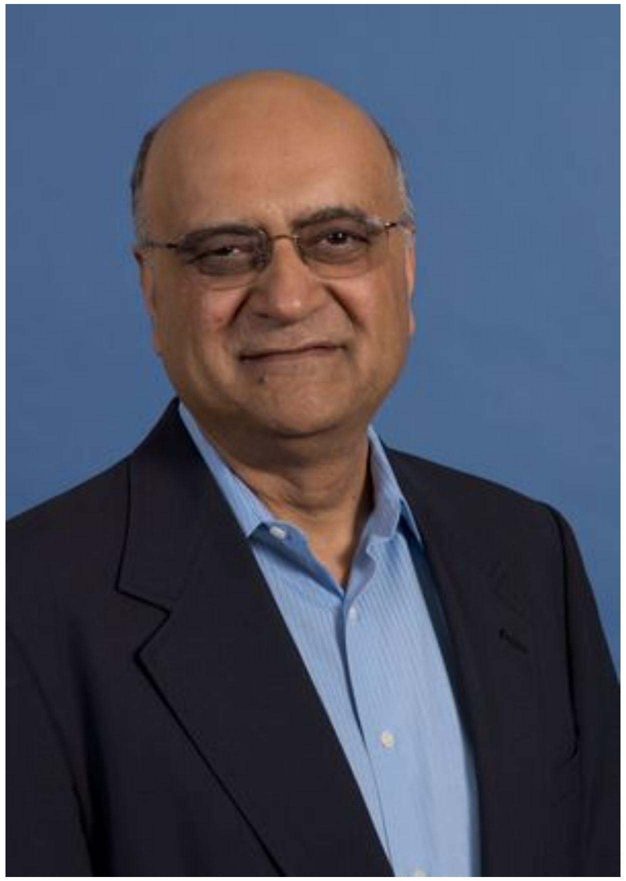
- Why use roles?
  - Fewer relationships to manage
    - potential decrease from $O(mn)$ to $O(m+n)$, where m is the number of users and n is the number of permissions
    - There are often more users than roles and more objects than roles
  - Roles are a useful level of abstraction
  - Organizations operate based on roles
  - Roles are likely to be more stable than the set of users and the set of resources
  - Roles can effectively implement the principle of least privilege
    - Finding the minimum set of necessary access rights is performed per role rather than per users

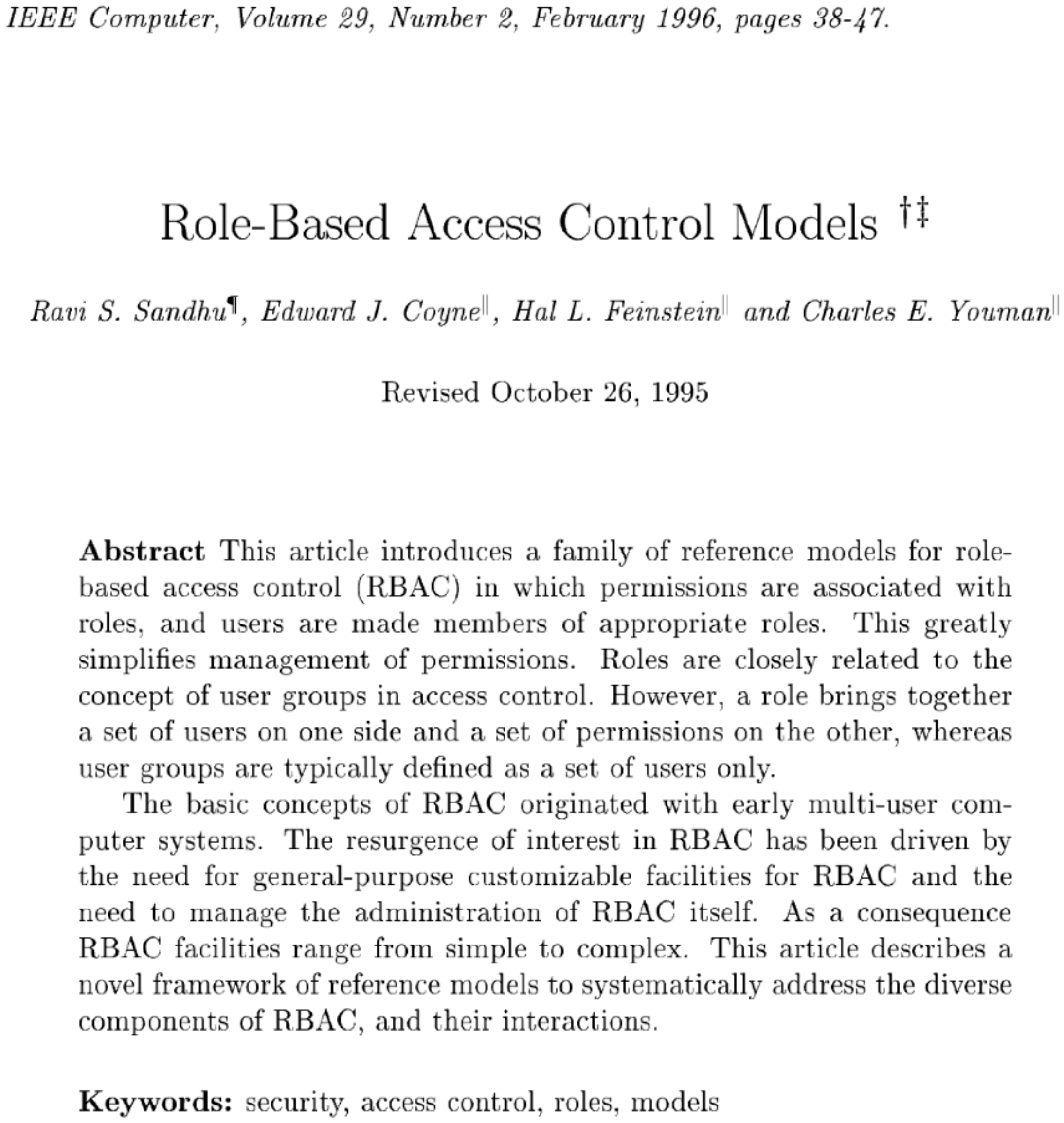
Groups vs. Roles

- How are roles different from groups?
  - A group is a collection of *users*, rather than a collection of *permissions*.
  - Another aspect of RBAC that distinguishes it from traditional group mechanisms is the concept of a session, which allows activation of a subset of roles assigned to a user.

History of RBAC

- Proposed in 1996 by Prof. Ravi S. Sandhu from UTSA





IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.

Role-Based Access Control Models [†‡]

Ravi S. Sandhu[¶], *Edward J. Coyne*[‖], *Hal L. Feinstein*[‖] *and Charles E. Youman*[‖]

Revised October 26, 1995

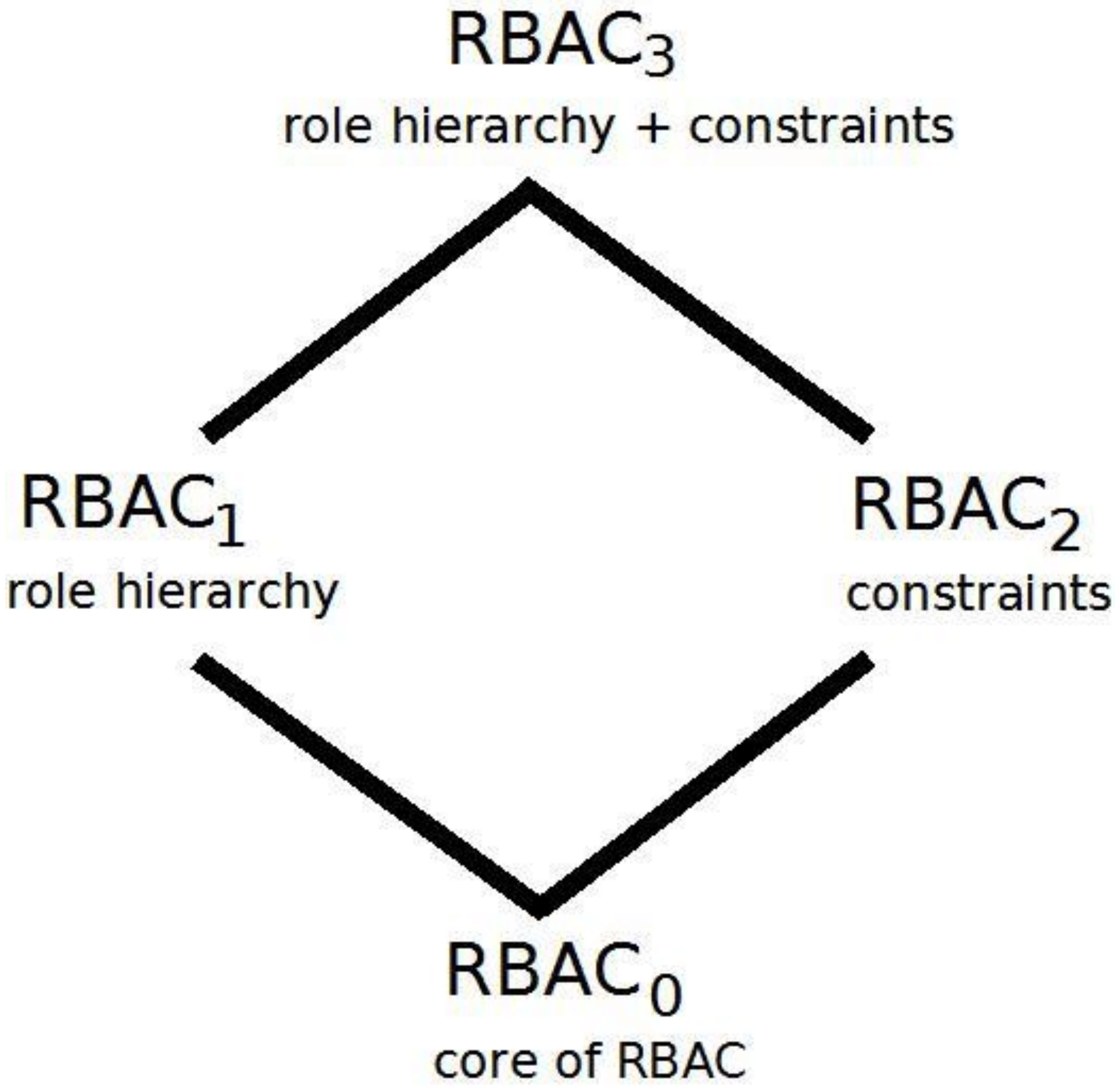
Abstract This article introduces a family of reference models for role-based access control (RBAC) in which permissions are associated with roles, and users are made members of appropriate roles. This greatly simplifies management of permissions. Roles are closely related to the concept of user groups in access control. However, a role brings together a set of users on one side and a set of permissions on the other, whereas user groups are typically defined as a set of users only.

The basic concepts of RBAC originated with early multi-user computer systems. The resurgence of interest in RBAC has been driven by the need for general-purpose customizable facilities for RBAC and the need to manage the administration of RBAC itself. As a consequence RBAC facilities range from simple to complex. This article describes a novel framework of reference models to systematically address the diverse components of RBAC, and their interactions.

Keywords: security, access control, roles, models

https://csrc.nist.gov/CSRC/media/Projects/Role-Based-Access-Control/documents/sandhu96.pdf

Family of RBAC Models [Sandhu et al. 1996]



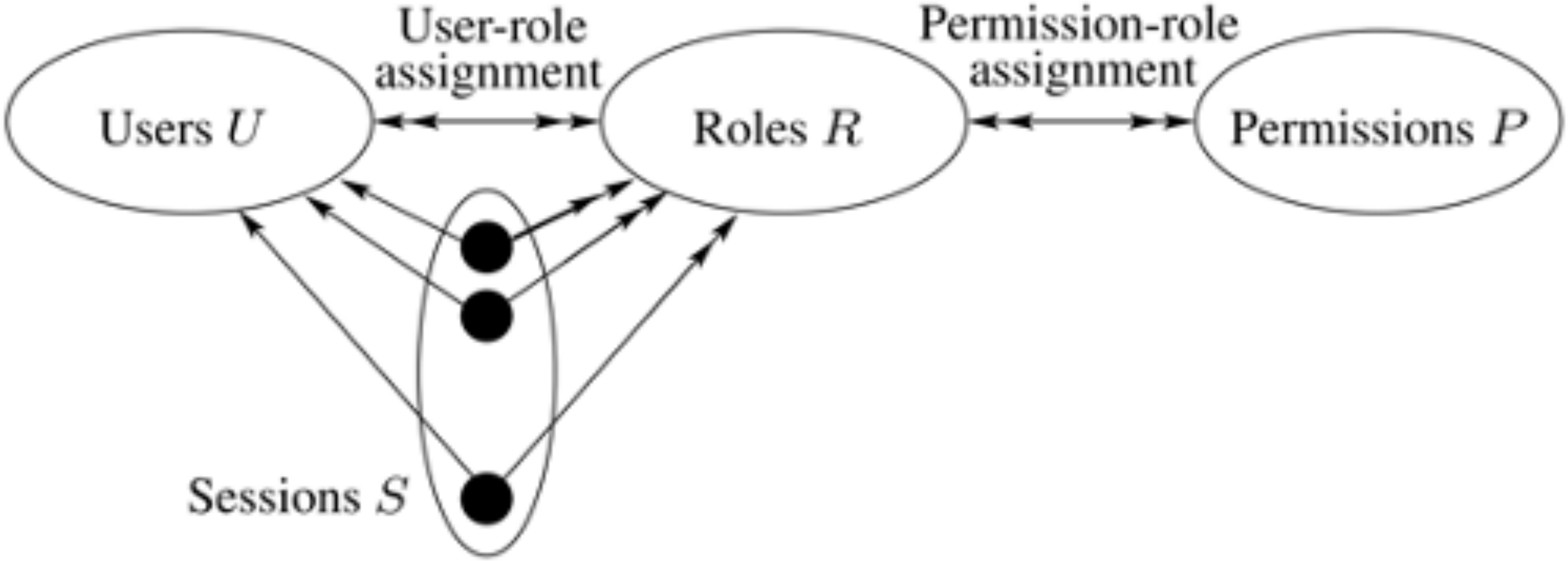
RBAC_0

- RBAC_0 contains four types of entities
  - Users U
  - Roles R
  - Permissions P
  - Sessions S

- User assignment (UA) is many-to-many $UA \subseteq U \times R$
- Permission assignment is many-to-many $PA \subseteq P \times R$
- Session activation
  - one-to-one for user: $S \mapsto U$
  - one-to-many for roles: $S \mapsto 2^R$

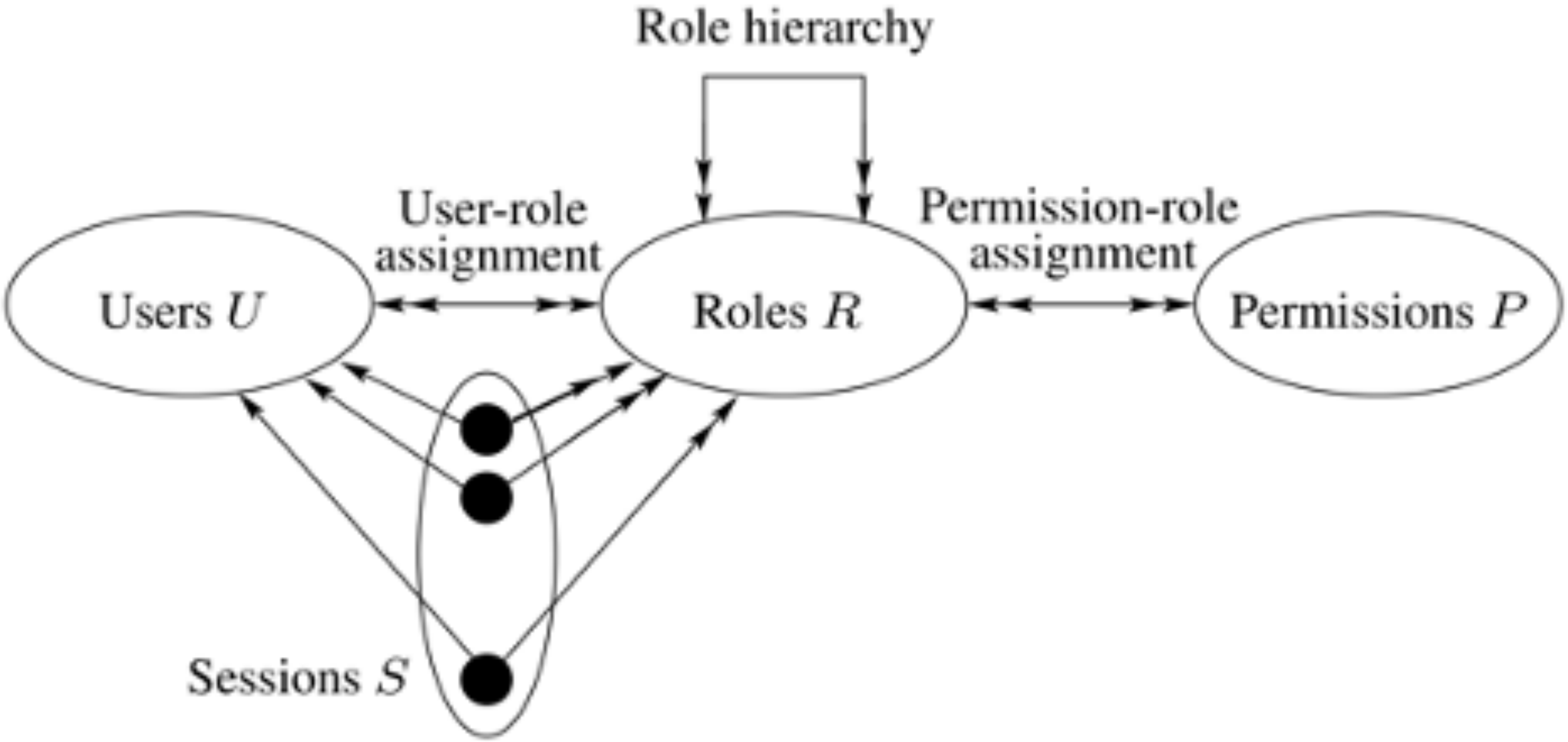
RBAC$_0$

- A session s must comply with UA and PA assignments
  - role$(s) \subseteq \{r \mid (\text{user}(s), r) \in UA\}$
  - permissions of session s are $\bigcup_{r \in \text{roles}(s)} \{p \mid (p, r) \in PA\}$



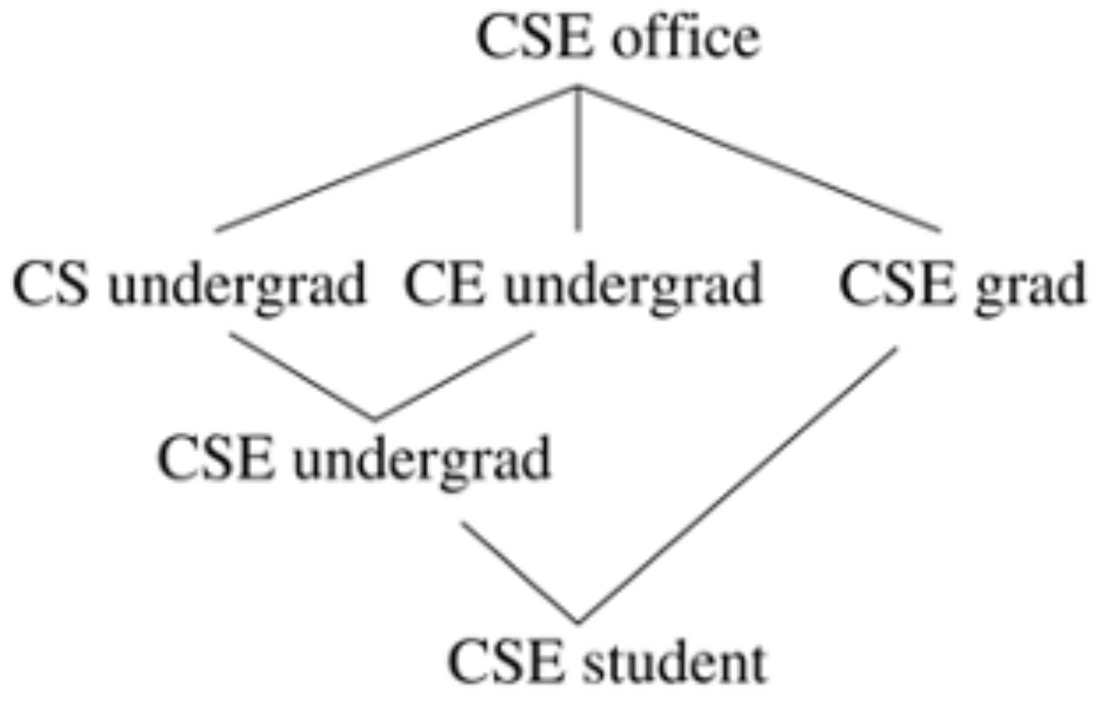
$RBAC_1$

- $RBAC_1$ enhances $RBAC_0$ with **role hierarchies**



RBAC$_1$

- **Role hierarchies** are based on the idea that subordinate job functions may have a subset of access rights of a superior job function
    - A role inherits access rights of its descendant roles
- **Example** of a role hierarchy

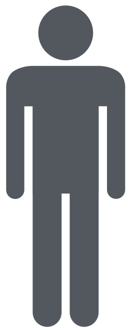


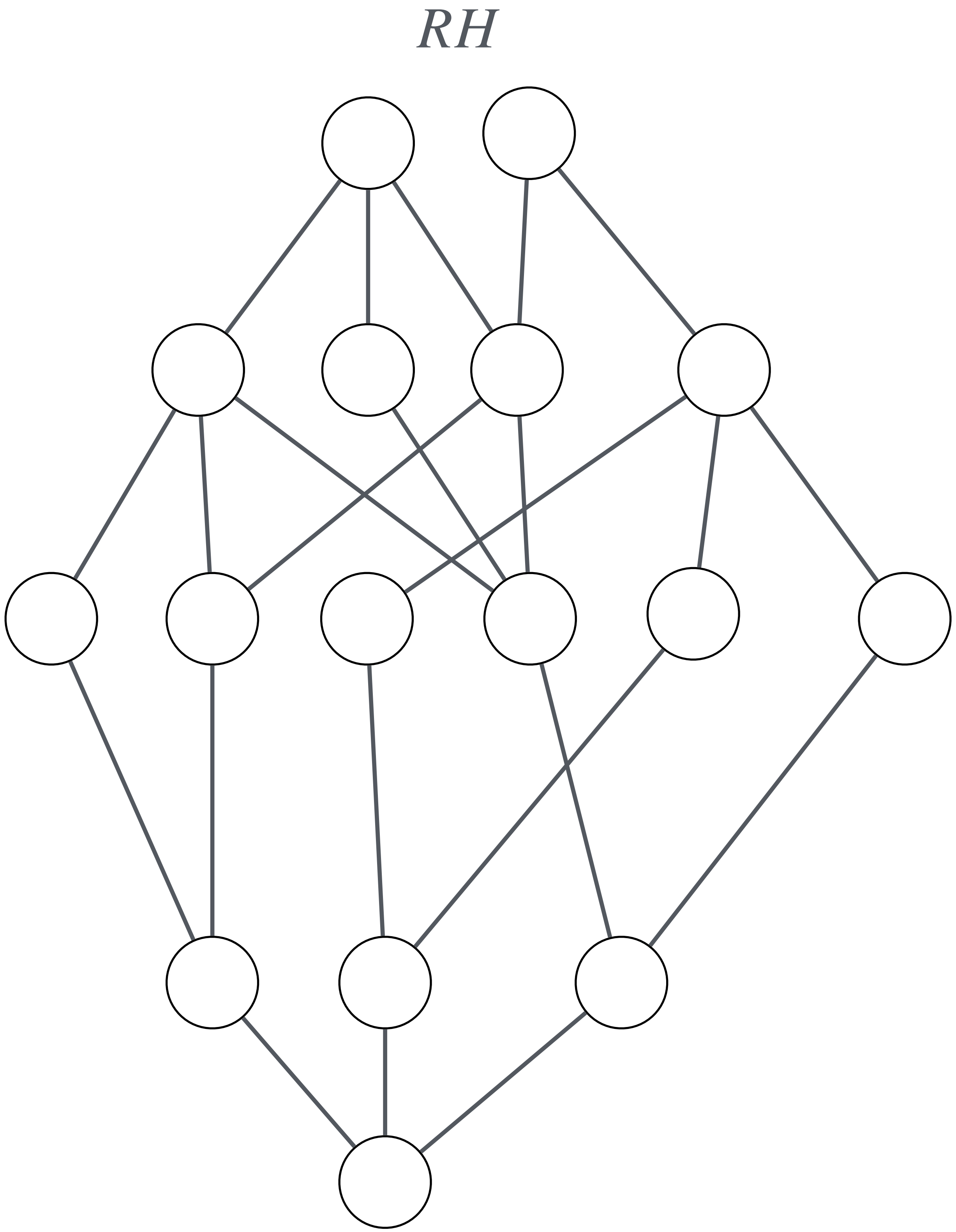
RBAC$_1$

- Formal model:
  - U, P, R, S, PA, UA are unchanged from RBAC$_0$
  - Role hierarchy $RH \subseteq R \times R$ is a partial order on R written as $\geq$
    - $r_1 \geq r_2$ means that r_1 is an ancestor of r_2
    - Partial order means that relationship between any two roles can be undefined
  - Requirements on session activation change
    - role$(s) \subseteq \{r \mid \exists r' \text{ s.t. } (r' \geq r) \,\&\, (\text{user}(s), r) \in UA\}$
    - session s has permissions $\bigcup_{r \in \text{roles}(s)} \{p \mid \exists r' \text{ s.t. } (r \geq r') \,\&\, (p, r) \in PA\}$

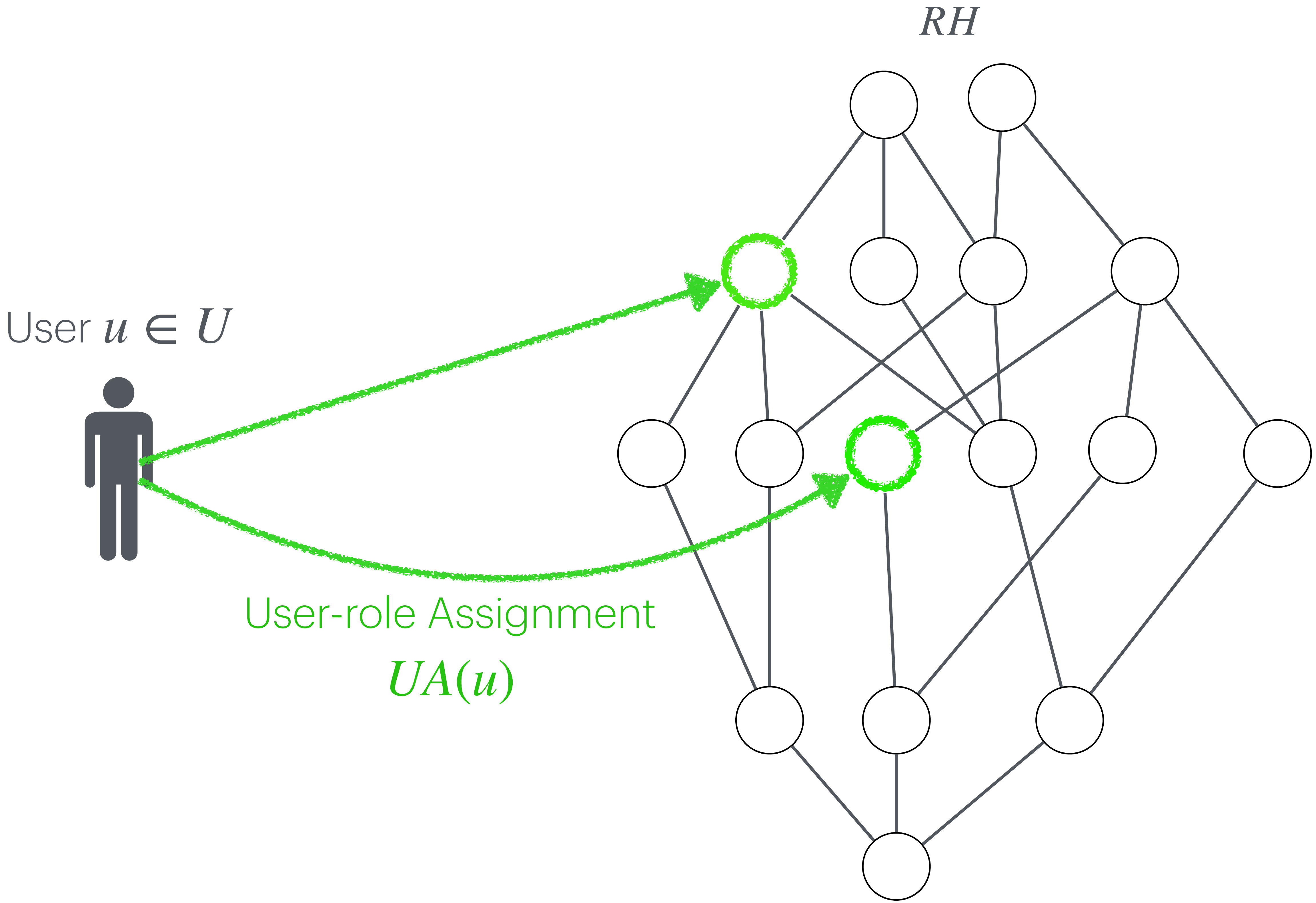
RBAC$_1$

User $u \in U$





RBAC$_1$



RH

User $u \in U$

User-role Assignment

$UA(u)$

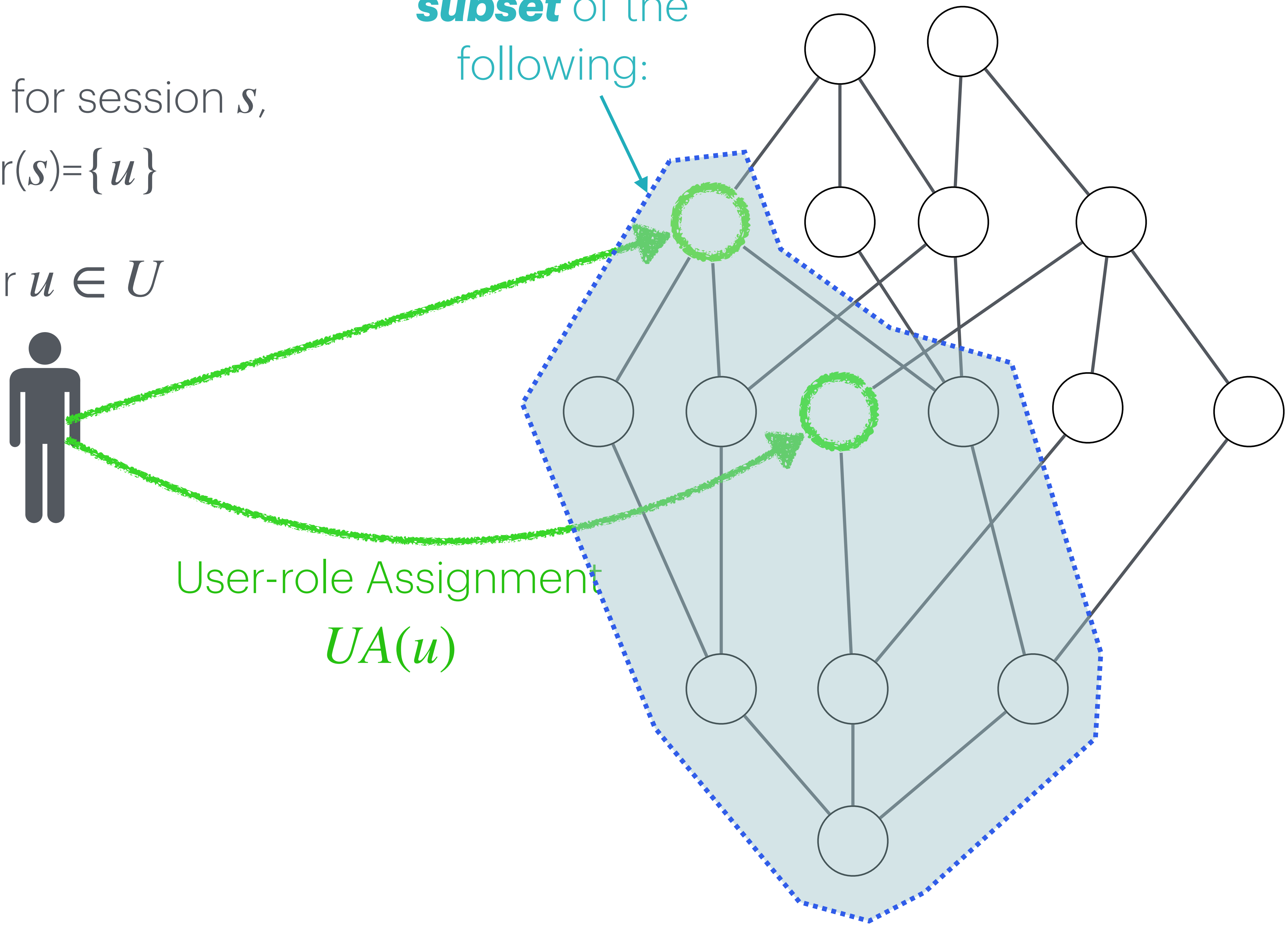
RBAC$_1$

Suppose for session s,

user(s)=$\{u\}$

User $u \in U$

role(s) is a ***subset*** of the following:



User-role Assignment

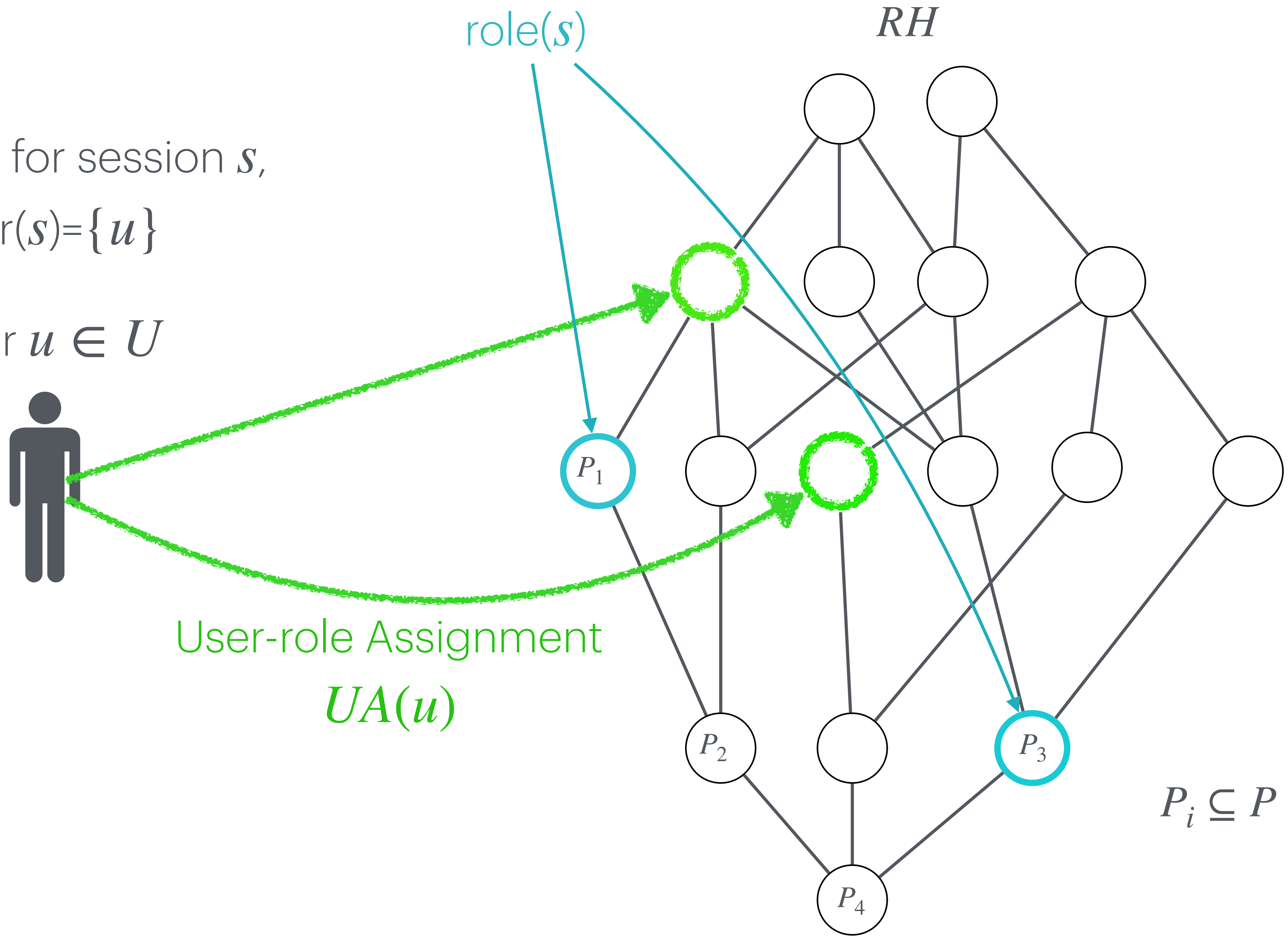
$UA(u)$

RBAC_1

Suppose for session s,

user(s)=$\{u\}$

User $u \in U$



role(s)

RH

P_1

P_2

P_3

P_4

User-role Assignment

$UA(u)$

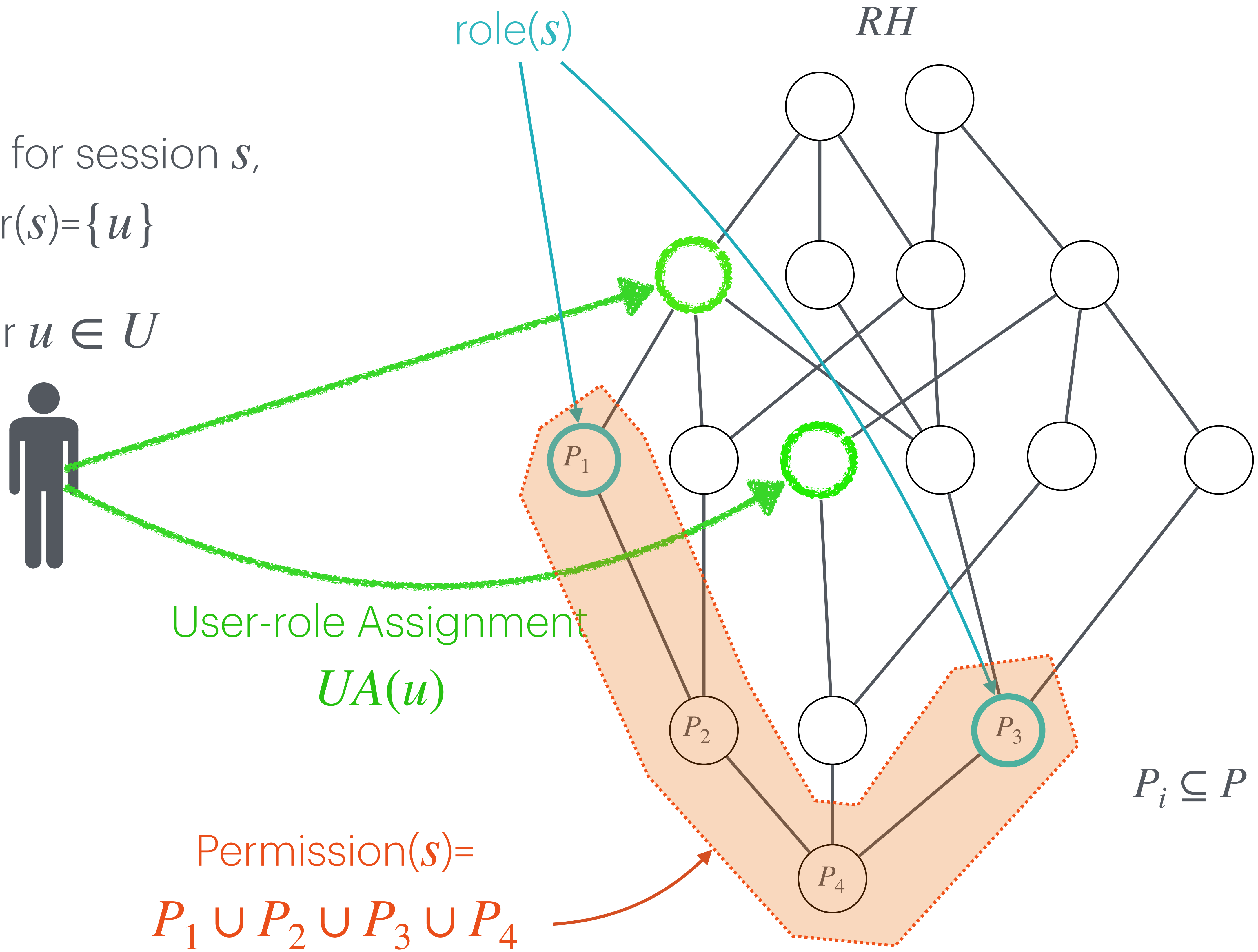
$P_i \subseteq P$

RBAC$_1$

Suppose for session s,

user(s)=$\{u\}$

User $u \in U$



role(s)

RH

User-role Assignment

$UA(u)$

$P_i \subseteq P$

Permission(s)=

$P_1 \cup P_2 \cup P_3 \cup P_4$

$RBAC_2$

- No formal model is specified for $RBAC_2$ that adds constraints to $RBAC_0$
- A constraint is a condition related to roles or a relationship defined on roles
- Types of constraints (Sandhu et al. 96)
  - Mutually exclusive roles
  - Cardinality constraints
  - Prerequisite constraints

Constraints in RBAC$_2$

- Mutually exclusive roles: a user can be assigned to only one role from a particular set of roles
  - Static exclusion
  - Dynamic exclusion
  - Such constraints support the separation of duties principle
- Prerequisite (or precondition) constraints: the prerequisite must be true before a user can be assigned to a particular role
  - a user can be assigned to role r_1 only if it is already assigned to another role r_2

RBAC in Use

- Products that use RBAC
  - Database management systems (e.g., Oracle)
  - Enterprise security management (e.g., IBM Tivoli Identity Manager)
  - Operating systems (e.g., Solaris OS, AIX)
- RBAC economic impact study (Tassey et al., NIST Report 2002)
  - Conducted by the Research Triangle Institute (RTI) based on interviews with software developers and companies that use RBAC
    - It estimated by 2006 30–50% of employees in service sector would be managed by RBAC systems (10–25% for non-service sectors)
    - It conservatively estimated the economic benefits of this degree of penetration through 2006 to be $671 million

Attribute-Based Access Control (ABAC)

Attribute-Based Access Control

- Attribute-based access control (ABAC) is a very recent mechanism for specifying and enforcing access control
  - Properties are specified in the form of *attributes*
  - Authorizations involve evaluating predicates on attributes
  - Conditions on properties of both the subject and resource can be enforced

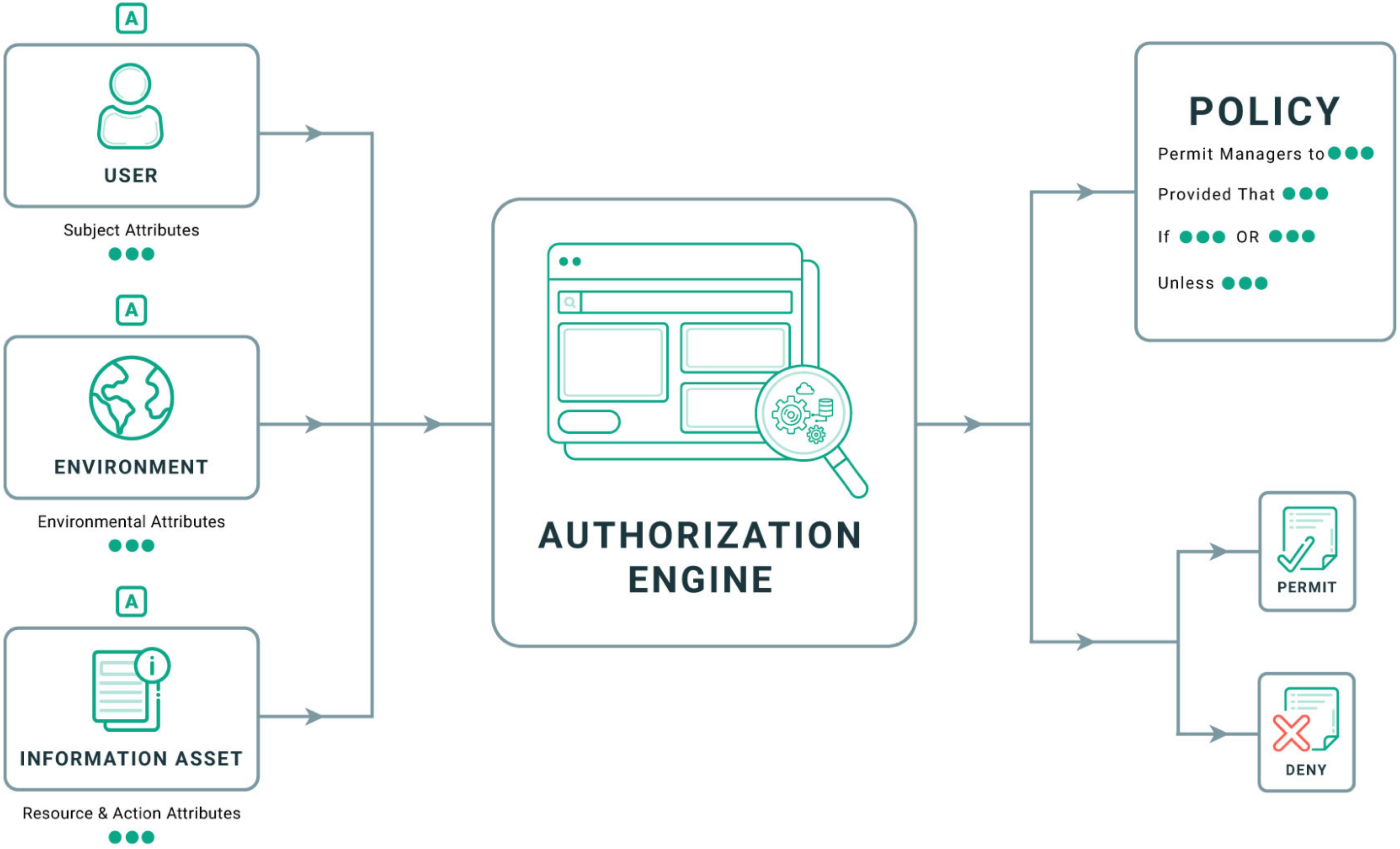
Attribute-Based Access Control

- ABAC provides a lot of flexibility in specifying rules and supports fine-grained access control
    - It is capable of enforcing DAC, MAC, and RBAC concepts
- This comes at a performance cost
    - It has seen the most success for web services and cloud computing where there is already a response delay
- There are three key elements in an ABAC model
    - Attributes, Policies, Architecture

Attribute-Based Access Control

- ABAC **attributes** are characteristics of subjects, objects, environment, and operations preassigned by an authority
- An ABAC model can have three types of attributes
  - Subject attributes
    - e.g., name, ID, job function, etc.
  - Object attributes
    - e.g., name, type, creation time, ownership information, etc.
  - Environment attributes
    - e.g., current date and time, network's security level, etc.

Attribute-Based Access Control



Attribute-Based Access Control

- An ABAC **policy** is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions
    - Typically written from the perspective of the object that needs protecting and the privileges available to subjects

Attribute-Based Access Control

- ABAC policies rules implement authorizations using **subject-object-environment** information (s, o, e)
  - There may not be explicit roles or groups and authorization decisions are instead made based on attributes
  - E.g., consider access to a database of movies
    - Everyone can access movies rated as G
    - Users of age $\geq$ 13 can access moved rated as PG-13
    - Users of age $\geq$ 17 can access movies rated as R
    - A policy might be written as $P_1(s, o, e)$: return (Age(s) ≥ 17 $\wedge$ Rating(o) $\in$ {R, PG-13, G}) $\vee$ (13 $\leq$ Age(s) $<$ 17 $\wedge$ Rating(o) $\in$ {PG-13, G}) $\vee$ (Age(s) $<$ 13 $\wedge$ Rating(o) $\in$ {G})

Attribute-Based Access Control

- ABAC policies can be combined into more complex rules
  - e.g., limit access to new releases to premium membership
    - $P_2(s, o, e)$: return (MemberType(s) = Premium) ∨ (MemberType(s) = Regular ∧ MovieType(o) = OldRelease)
  - Grant access if both rules are met
    - $P_3(s, o, e)$: return $P_1(s, o, e) \land P_2(s, o, e)$
    - The environment (e.g., the date) can be used for policies such as promotions

Attribute-Based Access Control

- ABAC **architecture** specifies how access control is enforced
- When a user submits an access request, the authorization decision is governed by
  - Access control policies
  - Subject attributes, Object attributes, Environmental attributes
- Contrast the above with ACLs in DAC
  - Broader and more dynamic.
  - Allows an unlimited number of attributes to be combined to satisfy any access control rule
  - ABAC systems are thus significantly more complex but expensive

Summary

- The choice of an access control model depends on the context
  - System requirements, security policies, etc.
  - Can use DAC, MAC, RBAC, ABAC, or other solutions
  - Have to consider costs of implementation, maintenance, and rule enforcement

Questions?