The GURA$_G$ Administrative Model for User and Group Attribute Assignment

Prof. Ravi Sandhu
Executive Director and Endowed Chair

10th International Conference on Network and System Security (NSS)
September 28-30, 2016

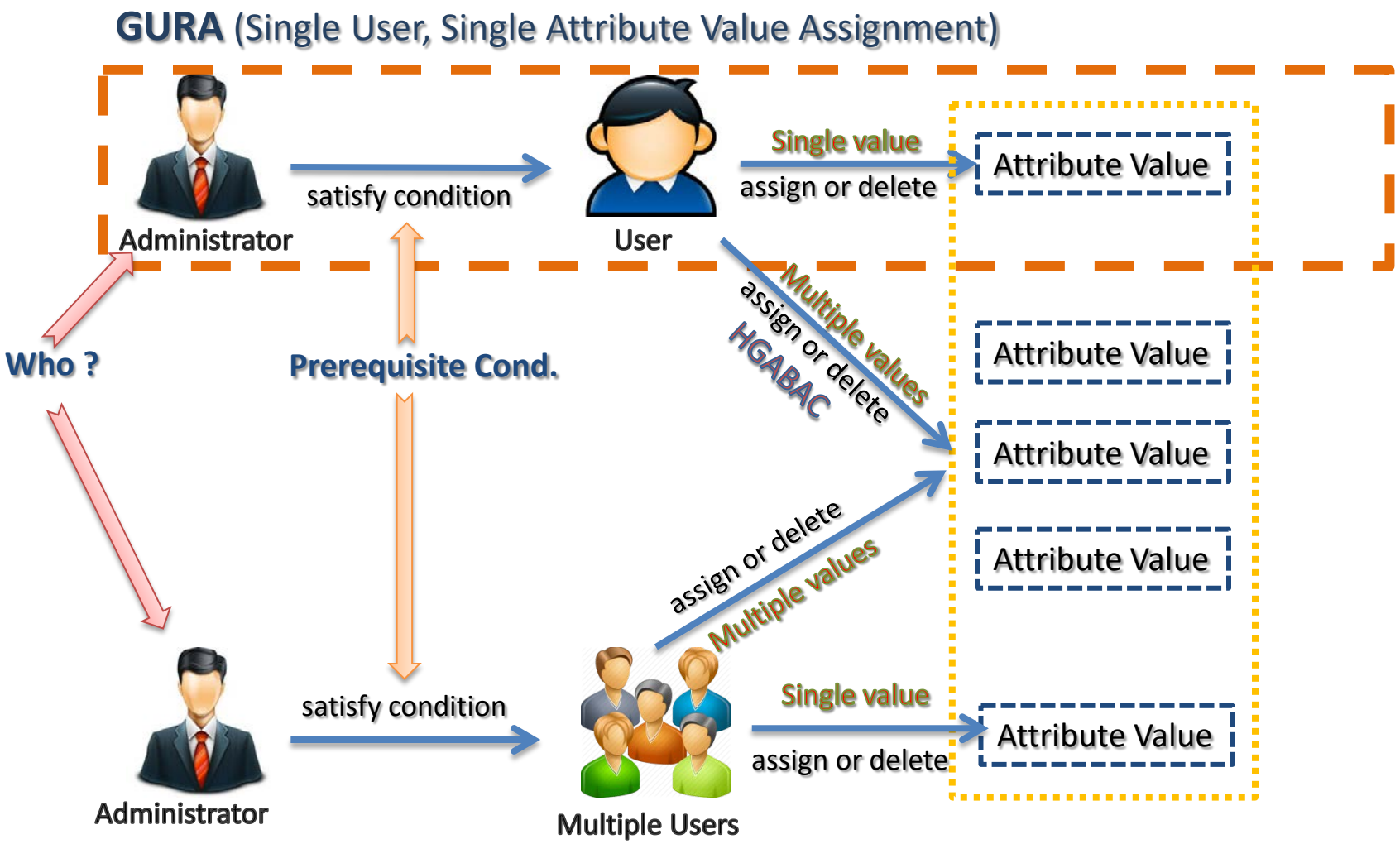
ravi.sandhu@utsa.edu
www.profsandhu.com

Maanak Gupta and Ravi Sandhu
Department of Computer Science

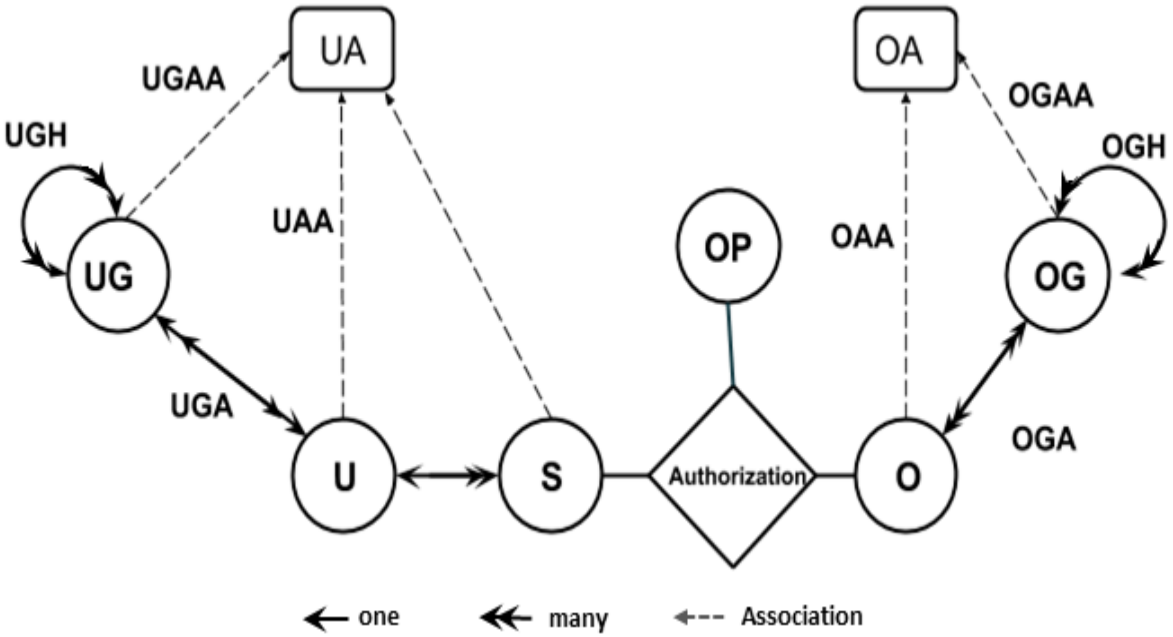
Attribute Based Access Control (ABAC)

- **Attribute Based Access Control**
  - requires attributes of entities to make access control decisions.
  - provides flexible and fine grained access control
  - needs attributes (characteristics of entities) to be assigned by security administrators before access policies can be enforced.
- **Several models have been developed**
  - $ABAC_{\alpha}$ model [DBSec12]
  - Attribute based encryption (ABE) [CCS06]
  - Logical Based Framework for ABAC [FMSE04]
  - Attributed based AC for web services [ICWS'05]
  - Guide to ABAC Definitions and Considerations [NIST SP 800-162]
  - etcetera!!

ABAC Administration



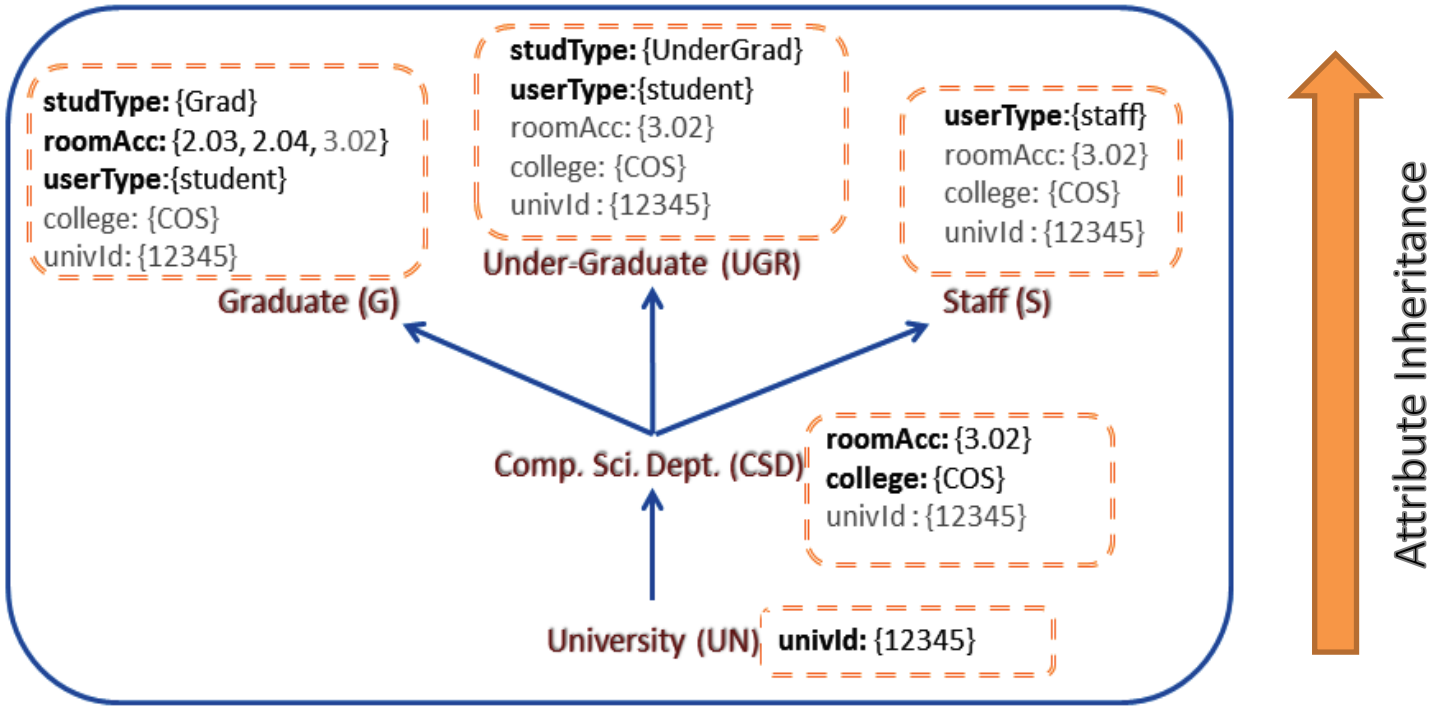
Redefined HGABAC



U: User
UG: User-Group
S: Subject
UA: User Attributes
O: Object
OG: Object-Group
OA: Object Attributes
OP: Operation (Actions)

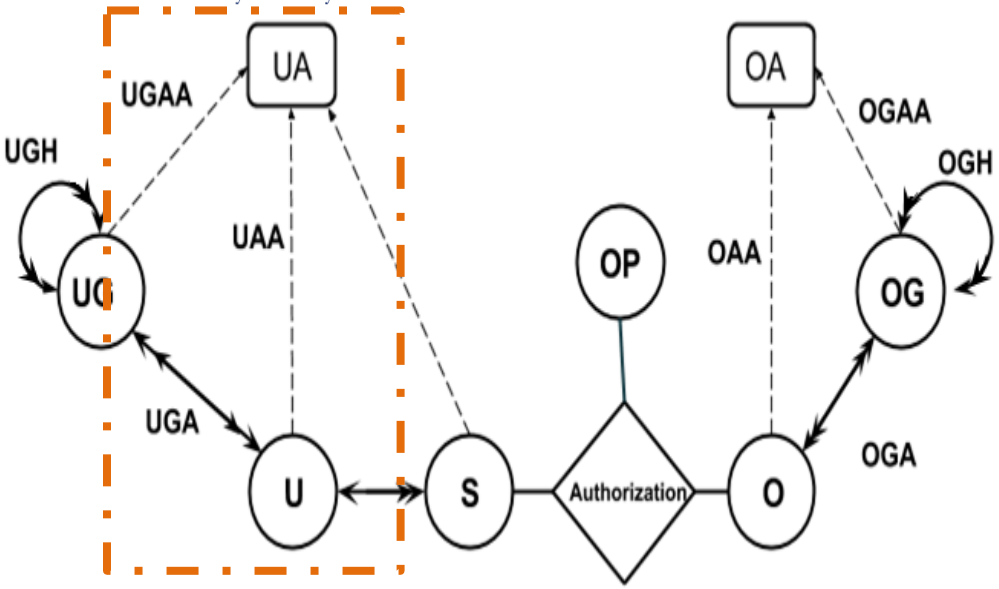
- [Servos et al] proposed Hierarchical Group and Attribute based Access Control (HGABAC) operational model
  - Introduces the notion of User and Object Groups
  - Core advantage is simplified administration of attributes
  - User and Objects are assigned set of attributes in one go as compared to single assignment at a time.

Example User-Group Hierarchy



- **Senior Groups inherit attributes from junior group**
  - Graduate group (G) is senior to CSD and UN
  - G inherits attributes from both CSD and UN
  - example: 'univId' and 'college' attribute for G inherited from UN and CSD
  - User assigned to group G will have direct attributes and attributes from G

$GURA_G$ Administrative Model



This paper proposes the first administration model for HGABAC model referred as **$GURA_G$.**

$GURA_G$ Sub Models

UAA: User Attribute Assignment

UGAA: User Group Attribute Assignment

UGA: User to User-Group Assignment

Administrative Relations

– User Attribute Assignment **(UAA)** & User-Group Attribute Assignment **(UGAA):**
For each att_u in UA,

$$canAdd_{att_u} \subseteq AR \times EXPR(UA) \times 2^{Range(att_u)}$$

$$canDelete_{att_u} \subseteq AR \times EXPR(UA) \times 2^{Range(att_u)}$$

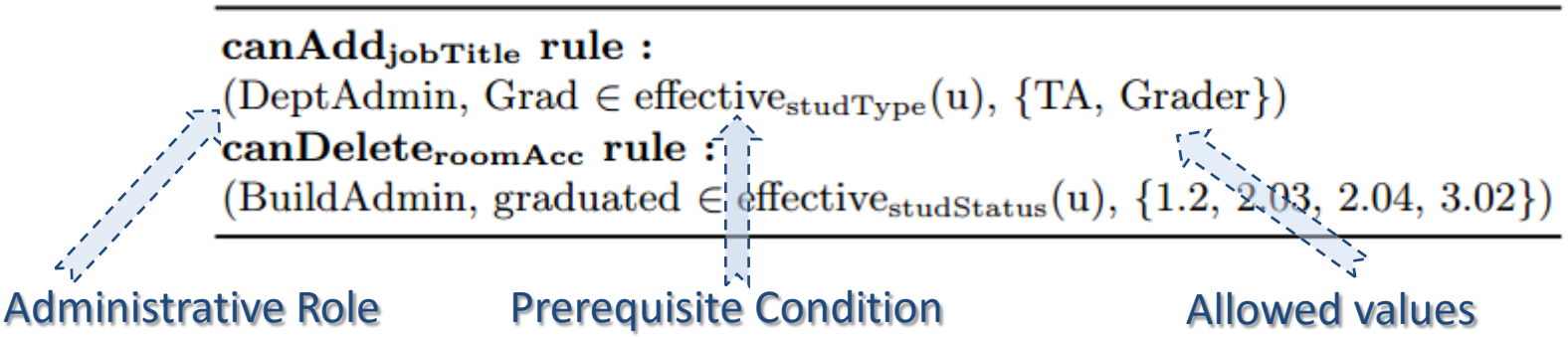
– User to User-Group Assignment **(UGA):**

$$canAssign \subseteq AR \times EXPR(UA \cup UG) \times 2^{UG}$$

$$canRemove \subseteq AR \times EXPR(UA \cup UG) \times 2^{UG}$$

User Attribute Assignment (UAA)

➢ Example UAA rules



Rule 1: Administrative Role DeptAdmin (or senior) can add any value in {TA, Grader} to user attribute 'jobTitle' if the user's 'studType' attribute includes 'Grad' value.

Common Policy Expression Language:

- $\alpha ::= \alpha \wedge \alpha \mid \alpha \vee \alpha \mid (\alpha) \mid \neg\alpha \mid \exists\, x \in set.\alpha \mid \forall\, x \in set.\alpha \mid set \triangle set \mid atomic \in set \mid atomic \notin set$
- $\triangle ::= \subset \mid \subseteq \mid \not\subseteq \mid \cap \mid \cup$
- $set ::= effective_{att_{u_i}}(s) \mid effective_{att_{o_i}}(o)$ for $att_{u_i} \in UA$, $att_{o_i} \in OA$
- $atomic ::= value$

EXPR(UA) in UAA:

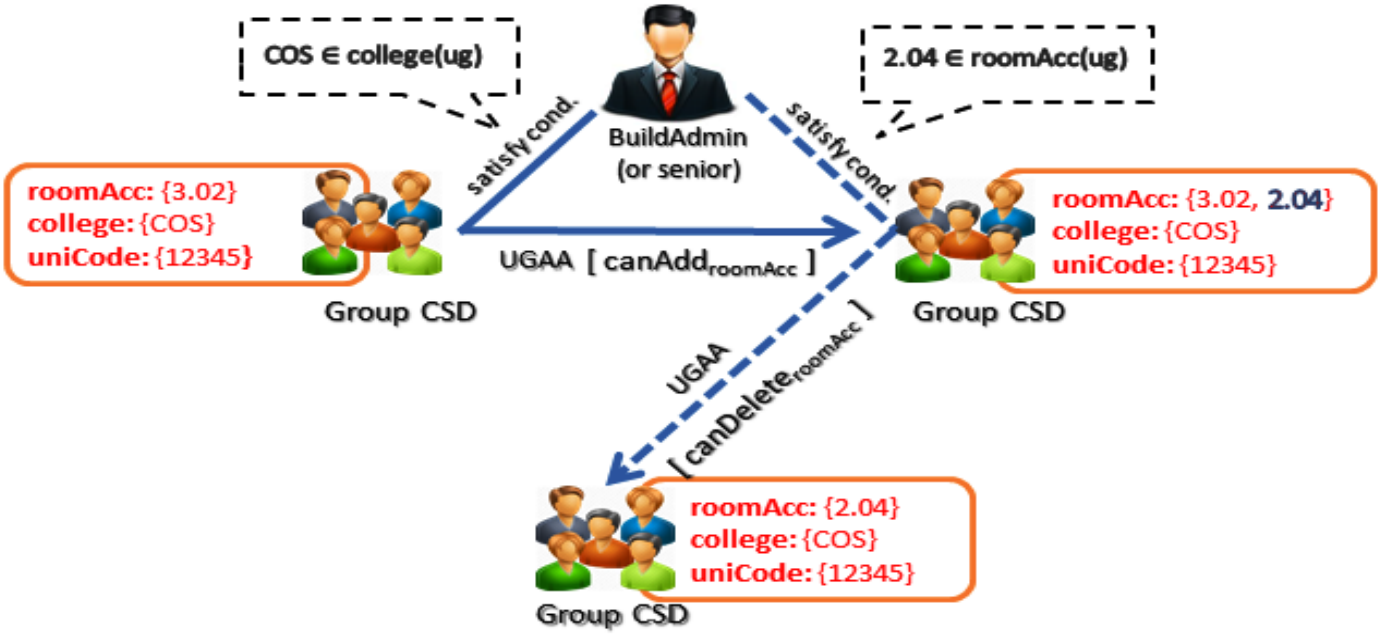
$set ::= att_{u_i}(u) \mid effective_{att_{u_i}}(u) \mid constantSet$ for $att_{u_i} \in UA$

$atomic ::= constantAtomic$

User Group Attribute Assignment (UGAA)

➢ Example UGAA rules

canAdd$_{roomAcc}$ rule: (BuildAdmin, COS ∈ college(ug), {2.04})
canAdd$_{skills}$ rule: (DeptAdmin, Grad ∈ studType(ug), {c++})
canDelete$_{roomAcc}$ rule: (BuildAdmin, 2.04 ∈ roomAcc(ug), {3.02})



EXPR(UA) in UGAA:

$$\text{set} ::= \text{att}_{u_i}(\text{ug}) \mid \text{effectiveUG}_{\text{att}_{u_i}}(\text{ug}) \mid \text{constantSet} \qquad \text{for } \text{att}_{u_i} \in \text{UA}$$
$$\text{atomic} ::= \text{constantAtomic}$$

User to User-Group Assignment (UGA)

Example UGA canAssign rules:

Admin Role	Prereq. Cond	AllowedGroups
DeptAdmin	$\{c, java\} \subseteq effective_{skills}(u) \wedge S \notin effectiveUg(u)$	{G,CSD}
StaffAdmin	$\{G,UGR\} \cap effectiveUg(u) = \emptyset \wedge Admin \in effective_{jobTitle}(u)$	{S}
DeptAdmin	$U \in directUg(u) \wedge 3.02 \in roomAcc(u) \wedge S \notin effectiveUg(u)$	{UGR,CSD}

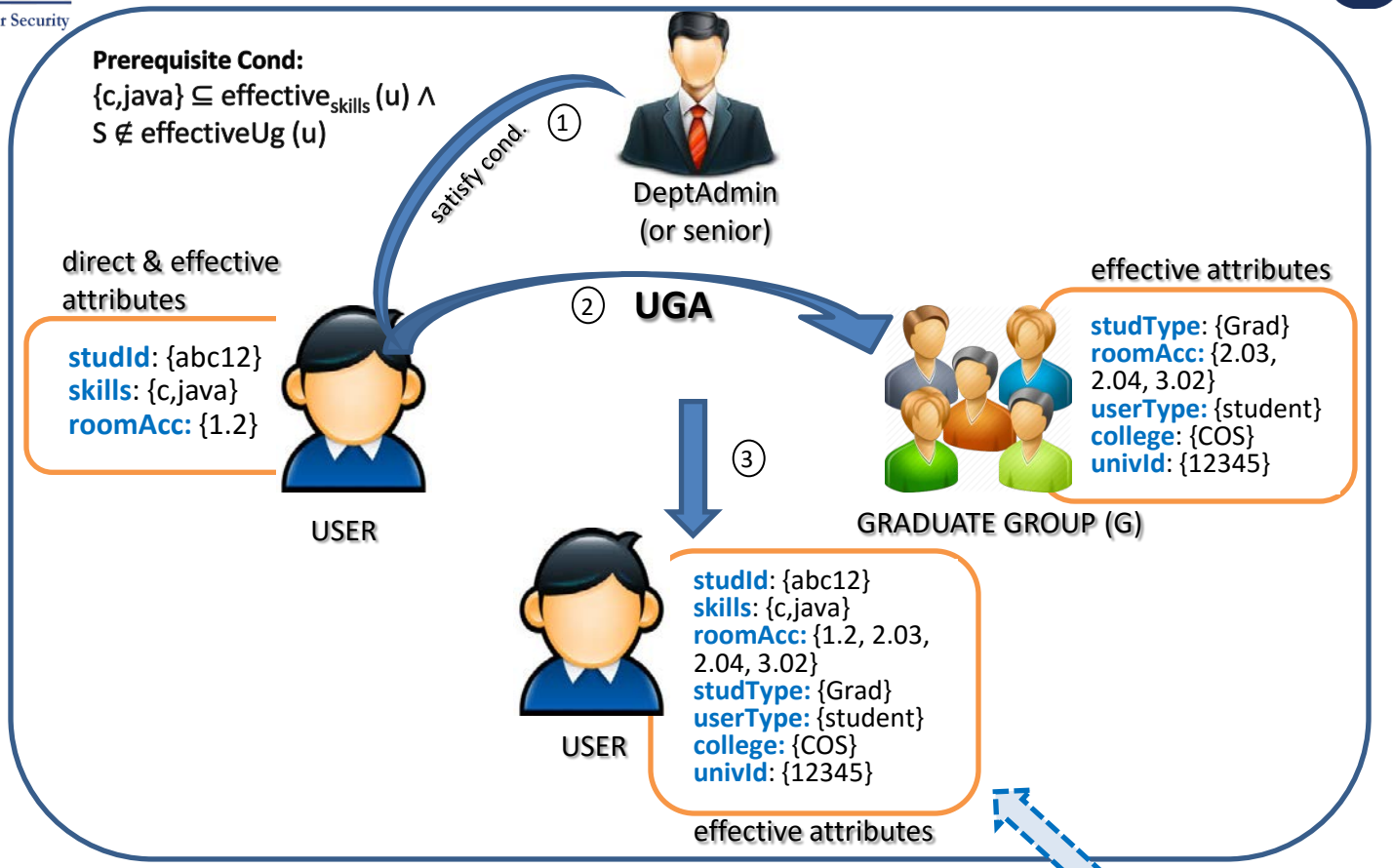
Example UGA canRemove rules:

Admin Role	Prereq. Cond	AllowedGroups
UniAdmin	$graduated \in effective_{studStatus}(u) \wedge \{G,UGR\} \cap effectiveUg(u) \neq \emptyset$	{G,UGR}
DeptAdmin	$COS \notin effective_{college}(u)$	{CSD}

EXPR(UA U UG) in UGA:

$$set ::= att_{u_i}(u) \mid effective_{att_{u_i}}(u) \mid directUg(u) \mid effectiveUg(u) \mid constantSet$$
$$atomic ::= constantAtomic$$
$$\text{where } effectiveUg(u) = directUg(u) \cup \left(\bigcup_{\forall ug_i \in directUg(u)} \{ug_j | ug_i \succeq_{ug} ug_j\} \right)$$

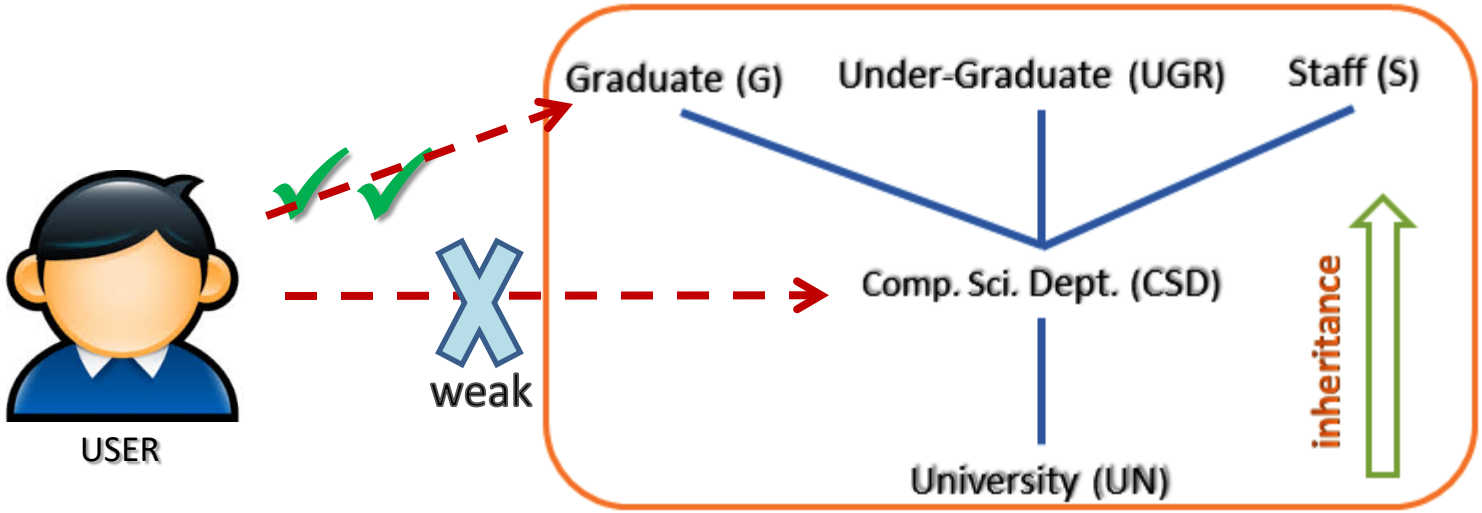
User to User-Group Assignment (UGA)



Here the user has been assigned set of attributes by group G membership, in lieu of single attribute assignment, making attribute administration easy.

GURA$_G$ Model Extensions

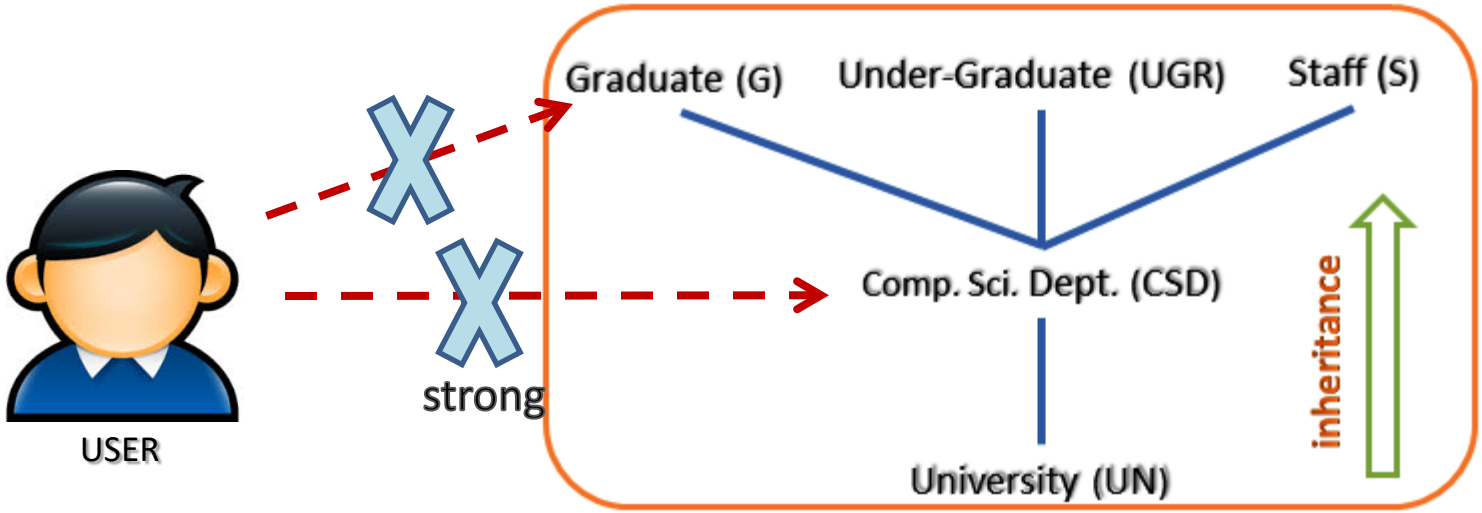
➢ Weak Removal versus Strong Removal



- **Weak Removal** will not impact implicit membership
  - After removal from CSD, user still inherits attribute of CSD through G.
- **Strong Removal** will remove both explicit and implicit memberships
  - User will be removed from G, if removed from CSD and authorized by rules.

GURA$_G$ Model Extensions

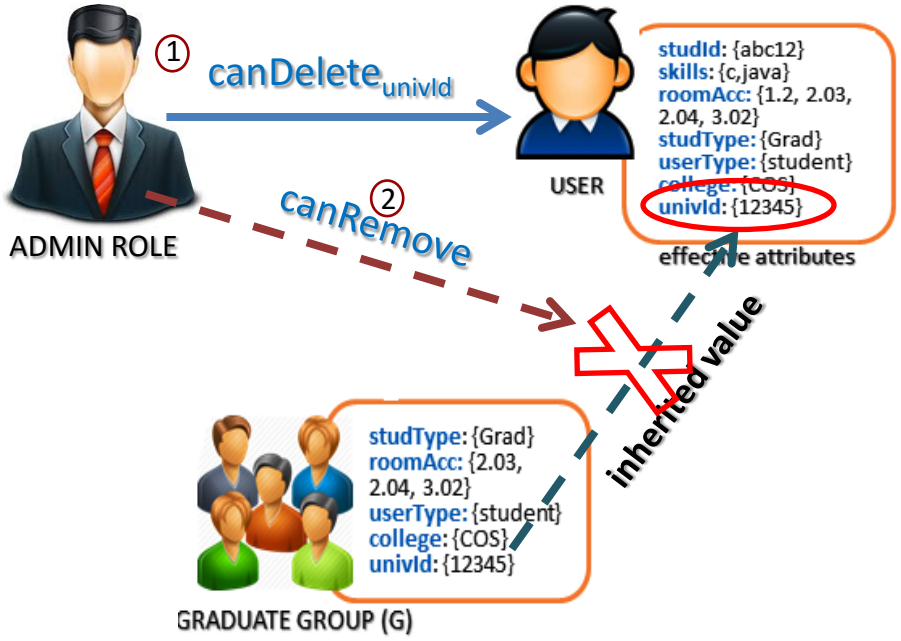
➢ Weak Removal versus Strong Removal



- **Weak Removal** will not impact implicit membership
  - After removal from CSD, user still inherits attribute of CSD through G.
- **Strong Removal** will remove both explicit and implicit memberships
  - User will be removed from G, if removed from CSD and authorized by rules.

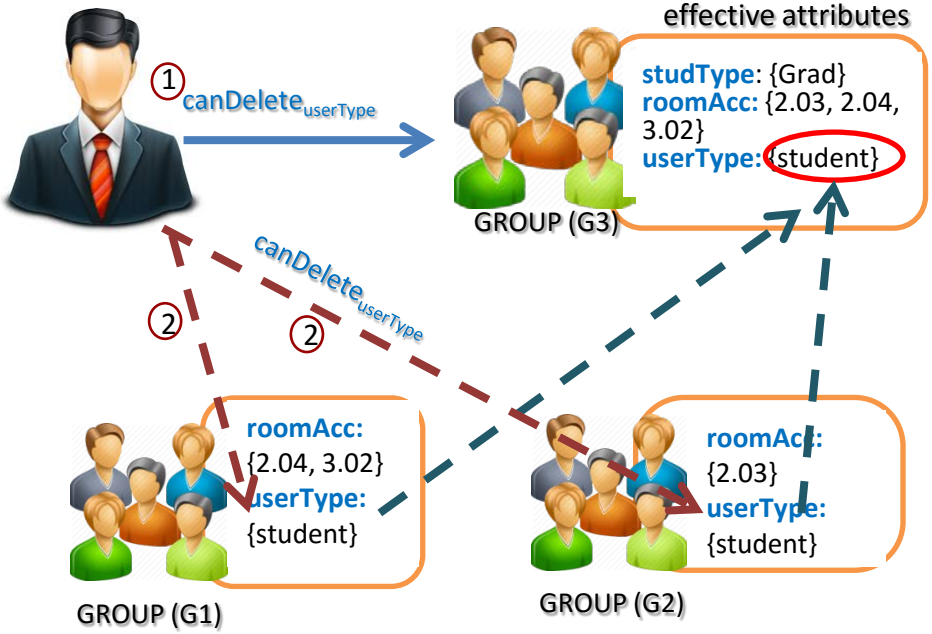
GURA$_G$ Model Extensions

➢ Inherited Value Deletion in User



Deleting an inherited value from a user will require to remove the membership of a user from all the user groups from where the value is inherited.

➢ Inherited Value Deletion in User Group



Deleting an inherited value from a user group will require the deletion of value from all the junior groups which have value directly assigned.

Note: Administrative Rules must exist to authorize operations.

Discussions and Future Work

Advantage:

- Simplified distributed attribute administration.
- RBAC advantage inherited.

Limitations:

- Cascading pre-assignment of attributes may lead to some values assignment not essentially required by the entity.
- UGA may require multiple pre-assignments of junior group to assign senior group, though the same inheritance can be achieved by senior group membership only.

Future Work:

- Reachability Analysis for $GURA_G$
- User and Object Group hierarchy administration.
- Attribute based User and Group attribute management.

Thank you!!

Any Questions??