On Feasibility of Attribute-aware Relationship-Based Access Control Policy Mining

Shuvra Chakraborty and Ravi Sandhu

Dept. of Computer Science and Institute for Cyber Security
University of Texas at San Antonio, TX 78249, USA

35th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec), July 19-20, 2021, Virtual Event.

Background

- *Access Control:* Legitimate users get legitimate access only
  - *ReBAC (Relationship-Based Access Control)*
  - *ABAC (Attribute-Based Access Control)*

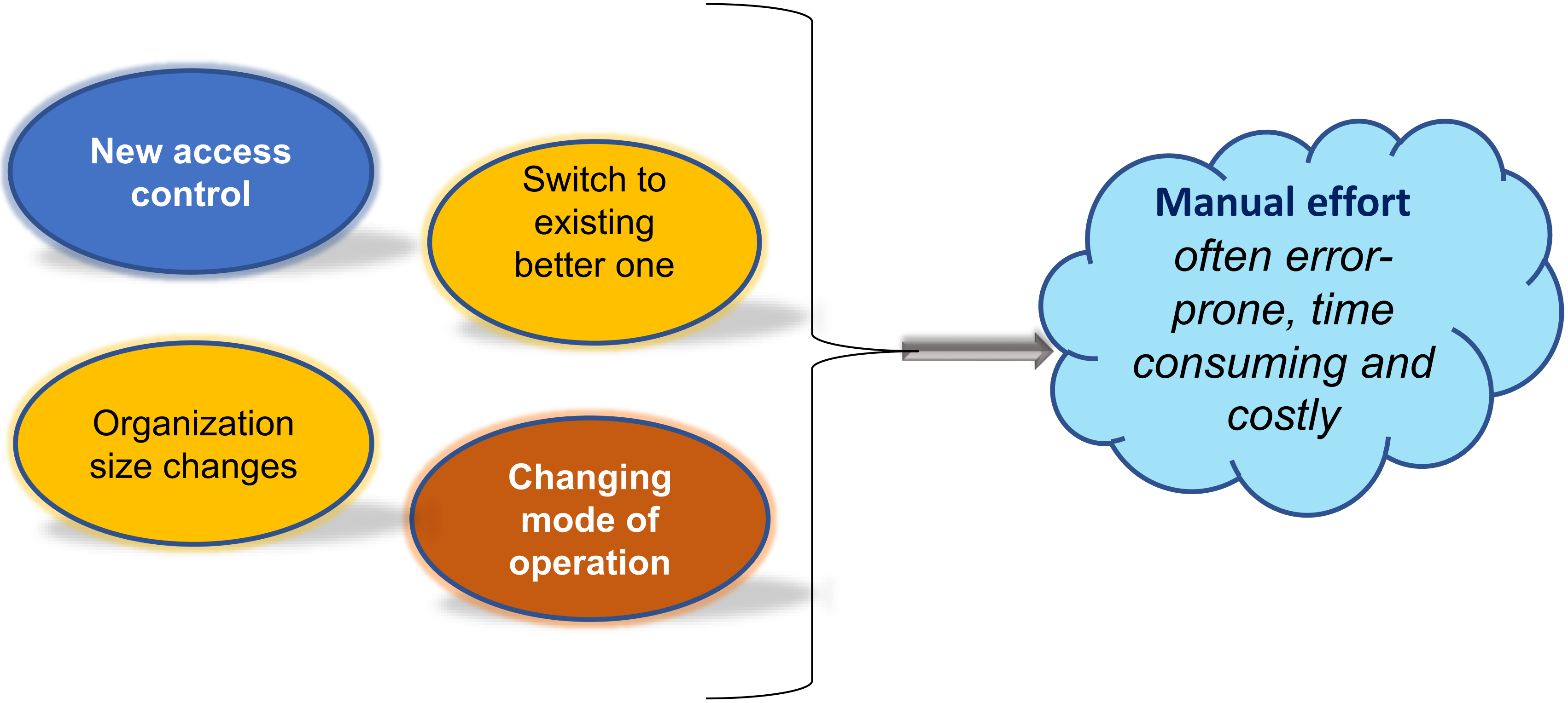
- *AReBAC ≡ Attribute-aware ReBAC*
  - Integrate attribute information with ReBAC
  - Makes policy generation more flexible and convenient
  - Attribute-aware Relationship Graph (ARG)

Assumption

- ARG where users(node) are connected(edge) where user and edge have attributes
- Each user and edge have corresponding user and edge attribute values, respectively
- Only user-to-user relationships are considered

Policy Mining

- **Problem:** migration from an existing access control model to another one


New access control
Switch to existing better one
Organization size changes
Changing mode of operation
Manual effort *often error-prone, time consuming and costly*

Is automation possible?

Problem Statement

The feasibility analysis of the AReBAC policy mining problem studies whether the migration process from a given authorization set to AReBAC policy is feasible or not under the set of imposed criteria:

- Attribute-aware Relationship Graph (ARG) is given
- AReBAC rule structure is given
- <u>Use of entity ID is not allowed</u>
  - <u>Existing literature allows ID</u>
- Equivalent set of AReBAC rules are required

- <u>Solution is guaranteed even if inconsistency arises</u>
  - <u>Infeasibility problem</u>

AReBAC Rule Structure

$Rule_{op}$ ::= $Rule_{op} \vee Rule_{op}$ | $pathRuleExpr$ | Attexp

$pathRuleExpr$ ::= $pathRuleExpr \wedge pathRuleExpr$ | ($pathLabelExpr$)

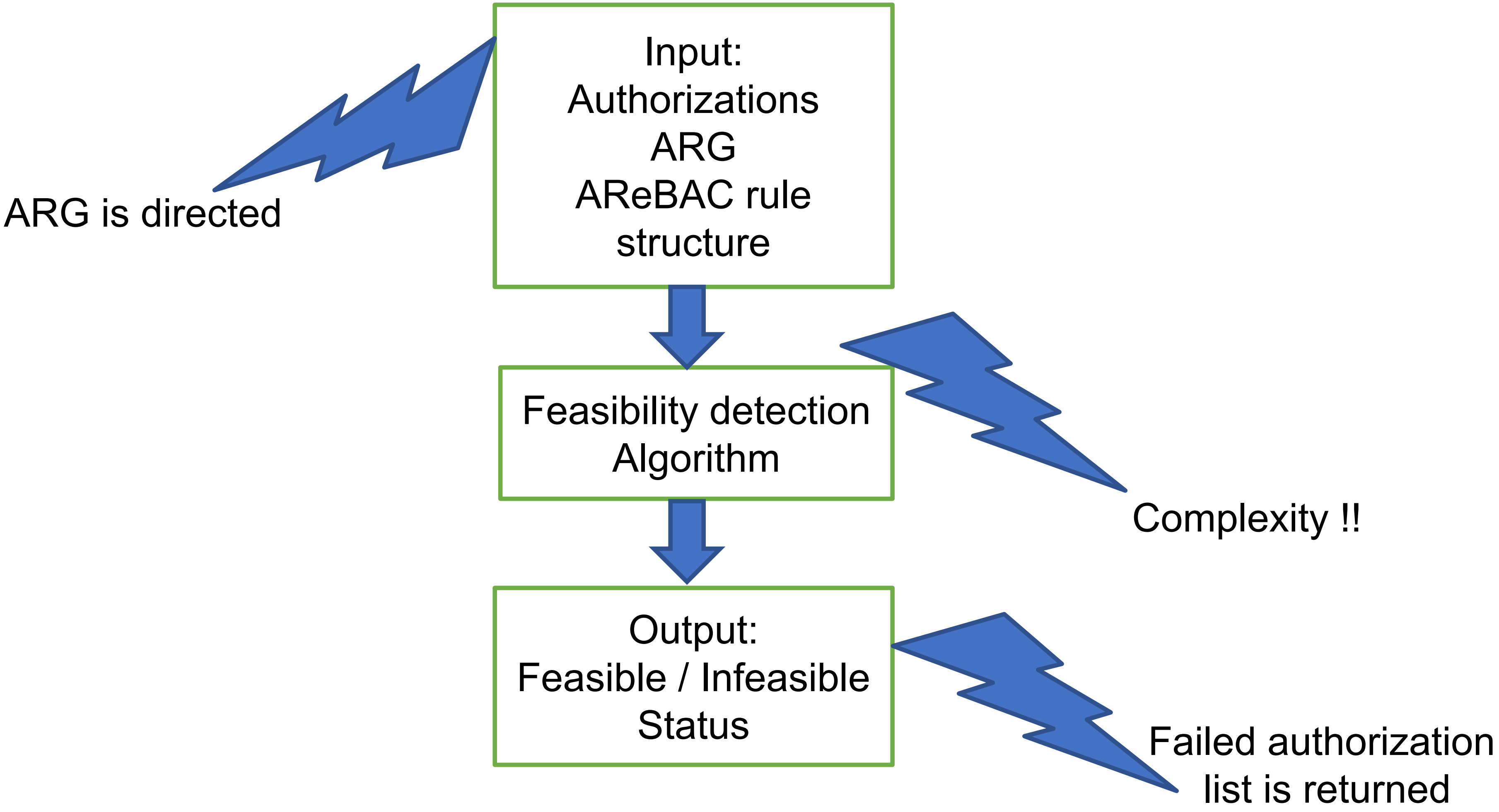
$pathLabelExpr$ ::= $pathLabelExpr.pathLabelExpr$ | $edge$Expr

Attexp ::= Attexp ∧ Attexp | uexp = value | vexp = value

edgeExp ::= edgeExp ∧ edgeExp | edgeuexp = value | edgevexp = value | edgeattexp = value

- Evaluation of access request (a, b, op)
  - Checks with user attribute values of a and b
  - If there exists simple path from a to b in ARG, Checks with them too!
  - The resulting boolean expression evalutes to true → grant, deny otherwise

Feasibility Detection



Input:
Authorizations
ARG
AReBAC rule structure

Feasibility detection Algorithm

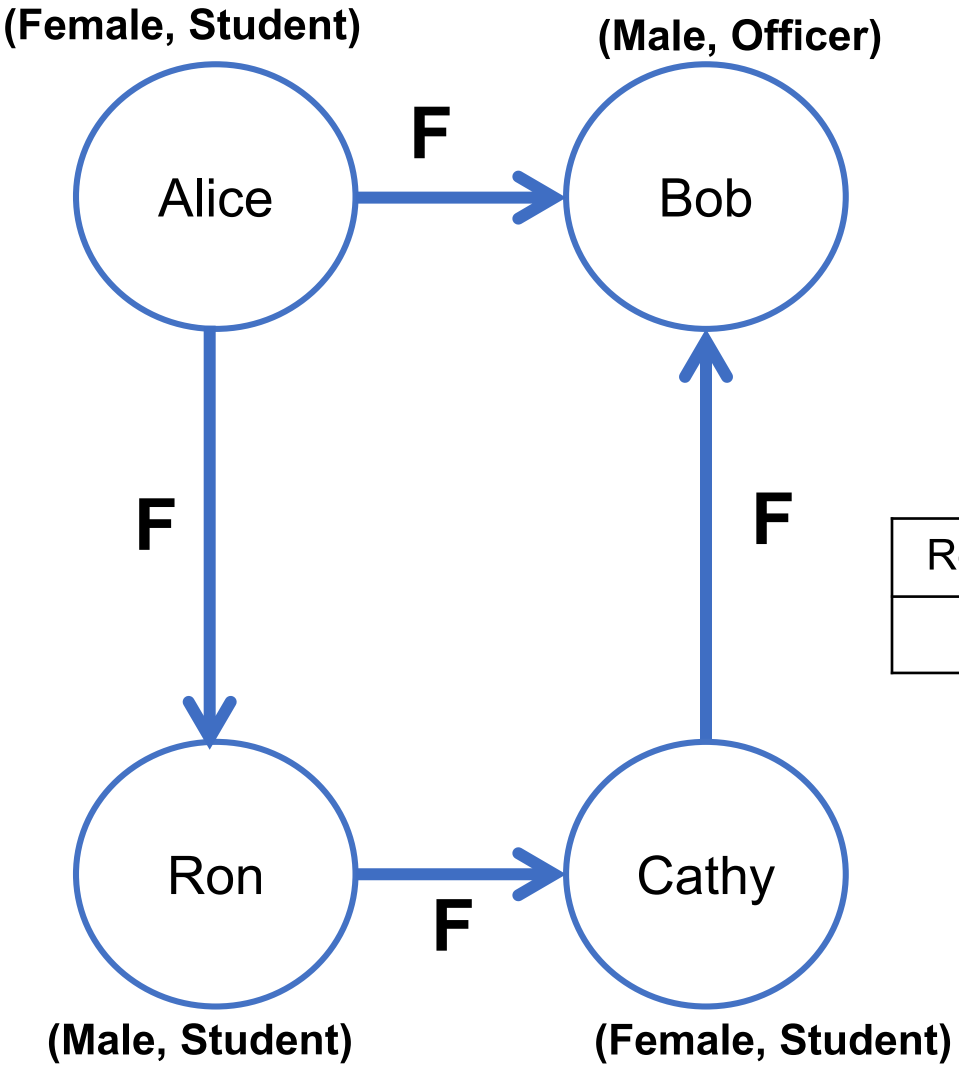
Output:
Feasible / Infeasible Status

ARG is directed

Complexity !!

Failed authorization list is returned

ARG Example



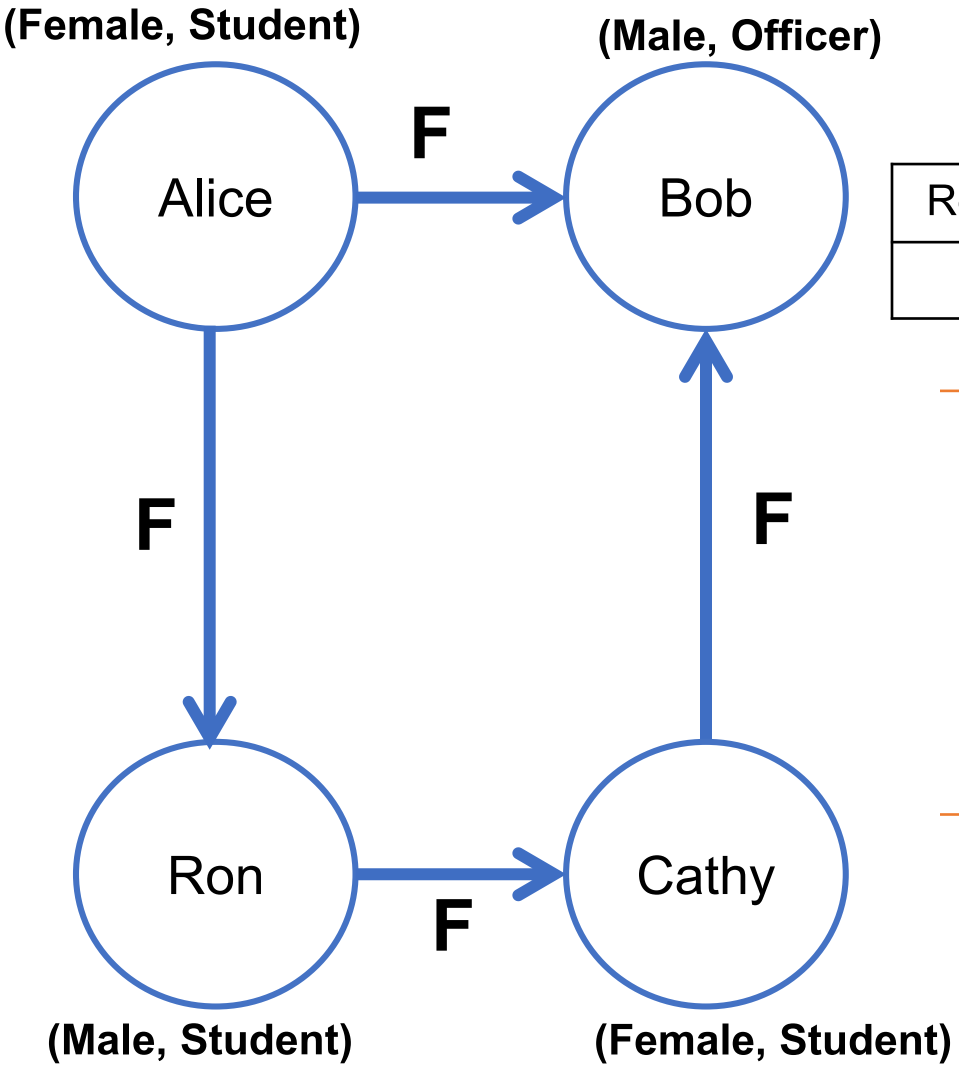
UA = {Gender, Profession}

EA = {Relation-type}

ReBAC	ABAC	AReBAC	AUTH
✗	✗	✓	(Alice, Ron, op)

Feasible

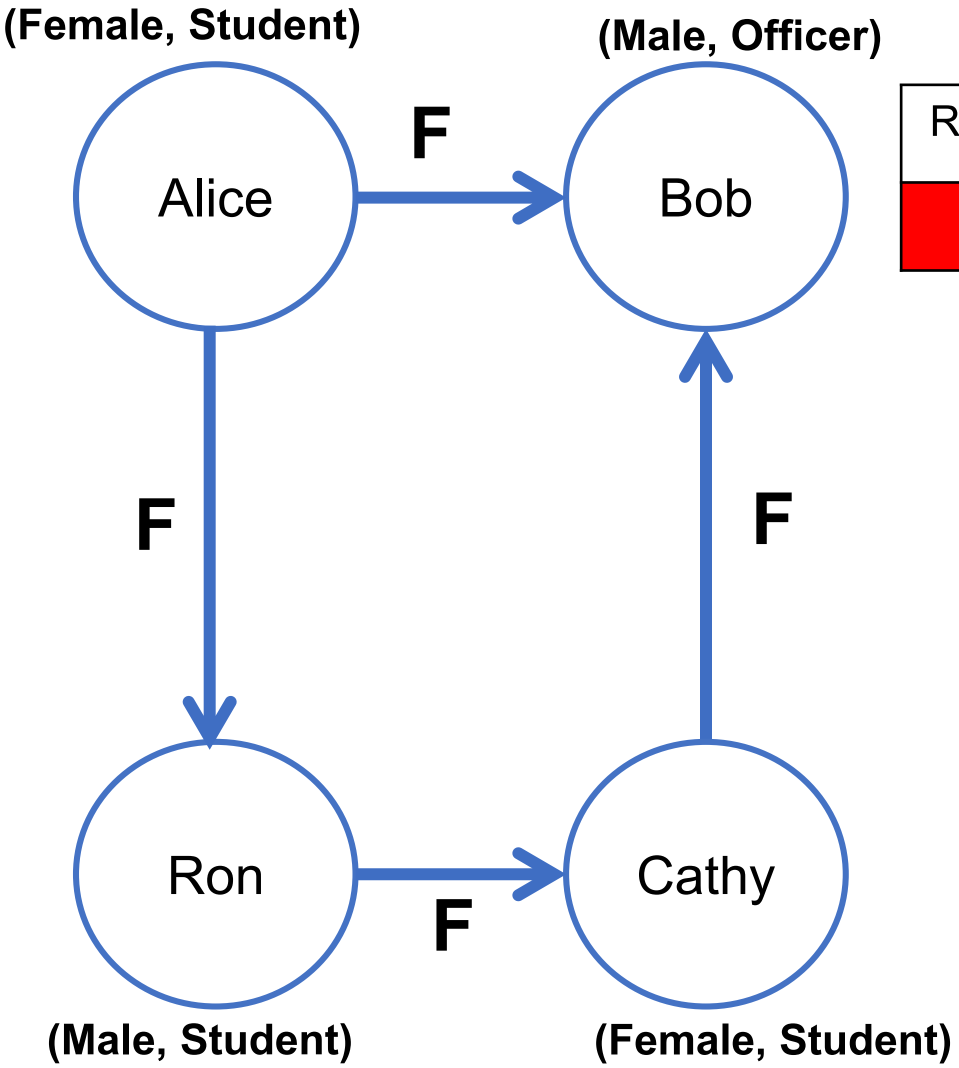
ARG Example



ReBAC	ABAC	AReBAC	AUTH
✗	✗	✓	(Alice, Ron, op)

Rule_{op} = (Gender(e.u) = Female ∧ Profession(e.u) = Student ∧ Relation-type(e) = F ∧ Gender(e.v) = Male ∧ Profession(e.v) = Student)

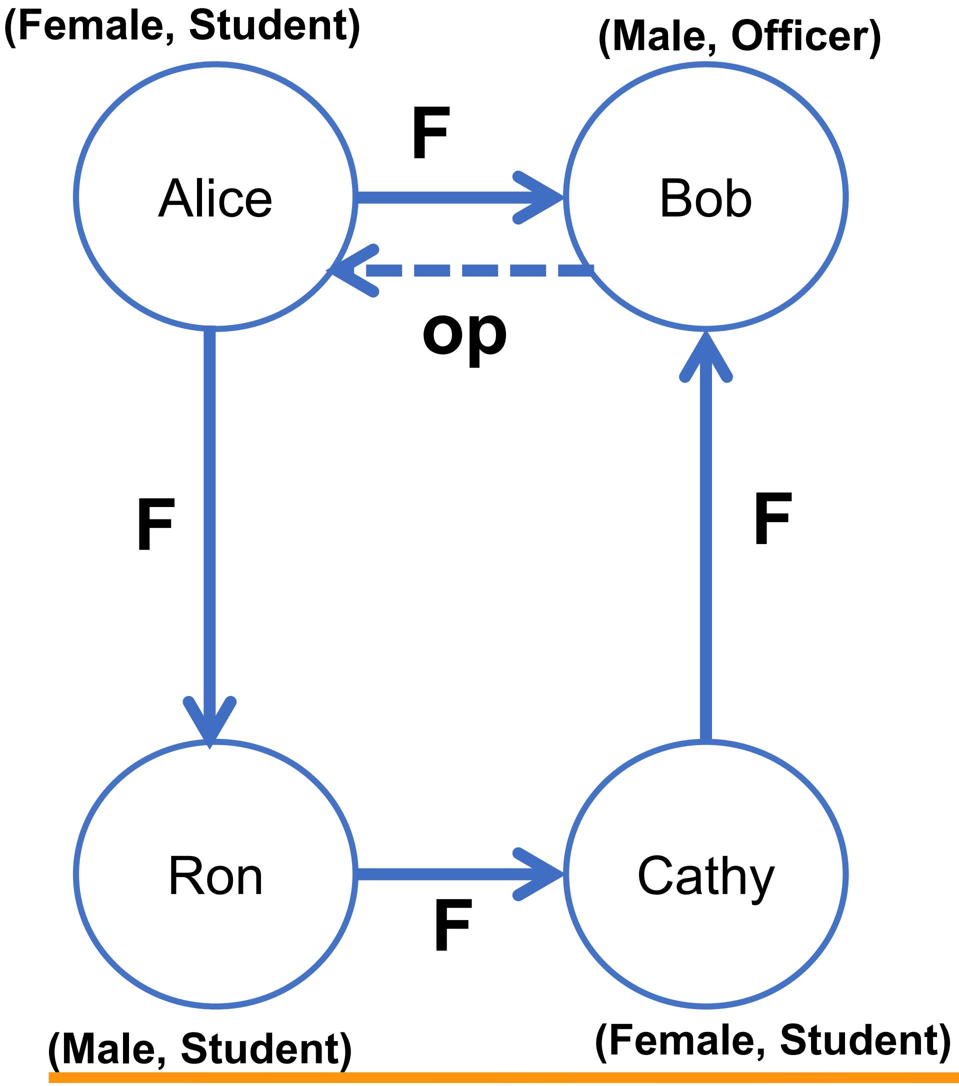
ARG Example



ReBAC	ABAC	AReBAC	AUTH
☒	☒	☒	(Bob, Alice, op)

Infeasible

Infeasibility Solution



Infeasible

(Bob, Alice, op)

Rule$_{op}$ = (Relation-type(e) = op)

Simple

Minimal edges not guaranteed

|Authorization| edges at worst!

Future Enhancement

- Complexity
- Inexact solution
- More path variations
- Cope up with changes in rule structures!
- Other infeasibility solutions
- Extend beyond user-user context

Acknowledgement

- This work is partially supported by NSF CREST Grant HRD-1736209

- Question/ Feedback