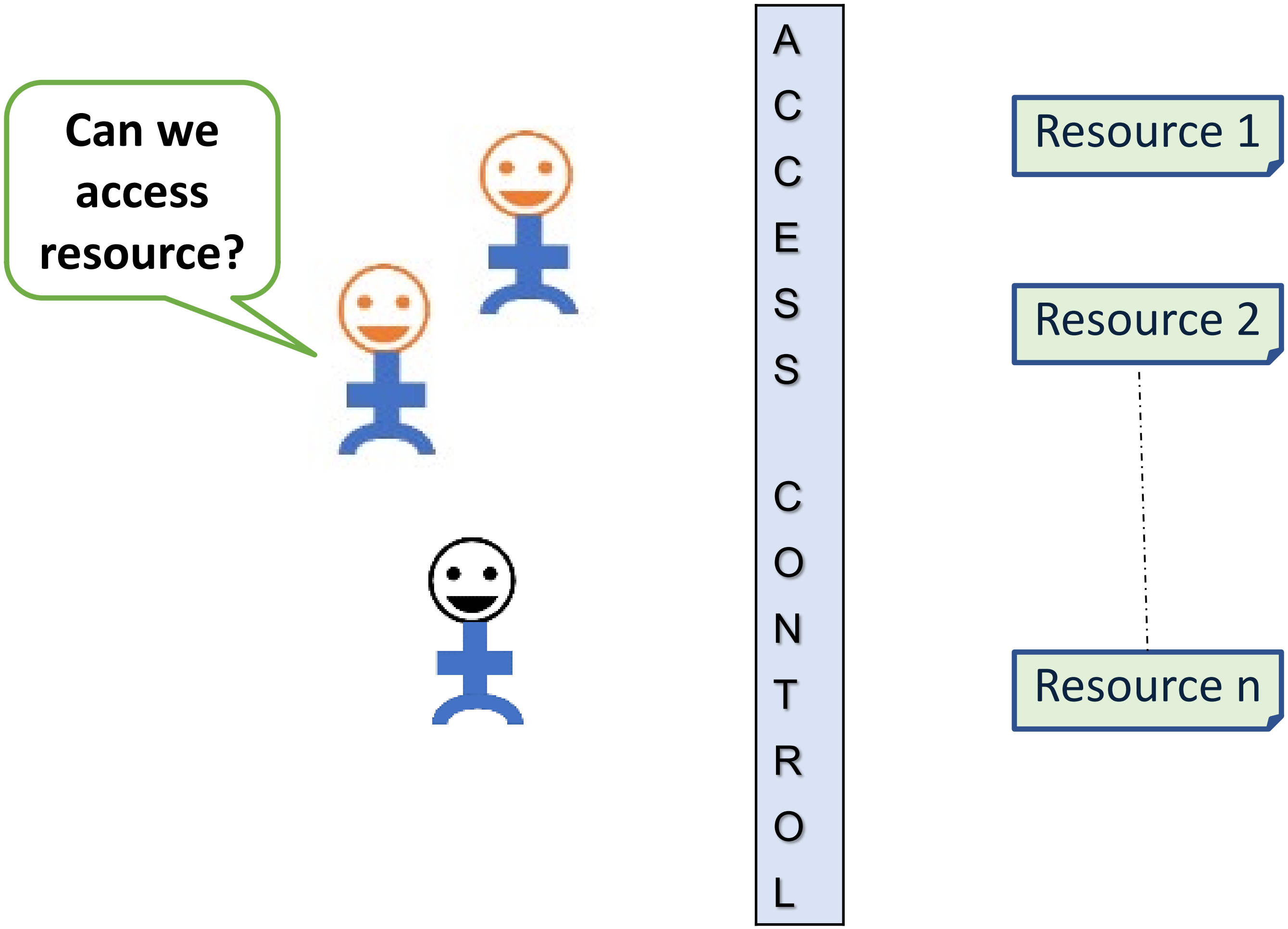
Formal Analysis of ReBAC Policy Mining Feasibility

Shuvra Chakraborty and Ravi Sandhu

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

11th ACM Conference on Data and Application Security and Privacy (CODASPY), April 26-28, 2021, Virtual Event.

Access Control



Legitimate users get legitimate access only

i.e., Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC)

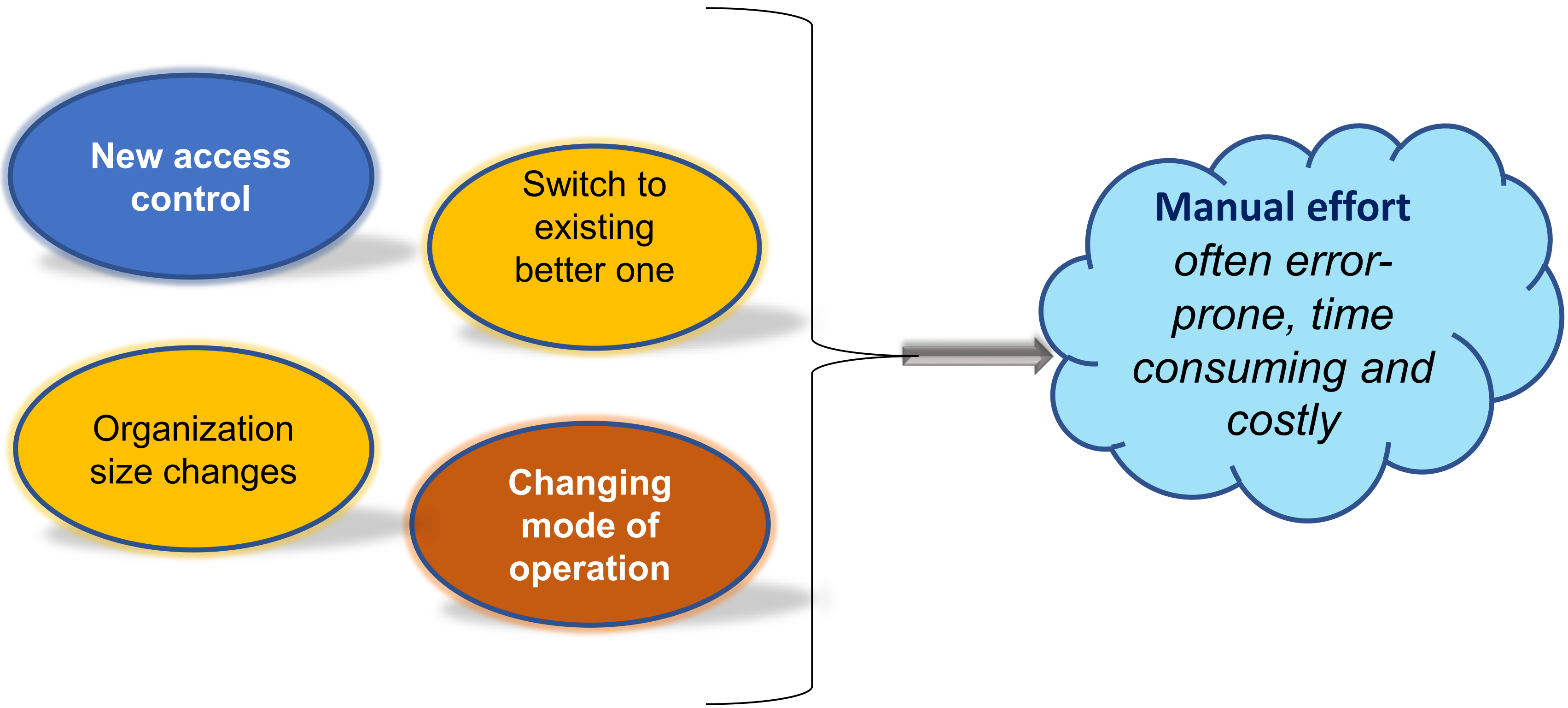
ReBAC

- *ReBAC ≡ Relationship-Based Access Control*
  - ReBAC expresses authorization in terms of various direct and indirect relationships amongst entities, most commonly between users
  - Access conditions are usually based on type, depth, or strength of relationships

- Assumption
  - Relationship Graph (RG) where users(node) are connected(edge) by social relationships(edge label). Each edge in the RG is labeled with a relation type
  - Only user-to-user relationships are considered

Policy Mining

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



Is automation possible?

Policy Mining Cont.

Access Control List / Log / RBAC + Supporting attribute data → ABAC policy mining

Access Control List + Supporting Relationship data → ReBAC policy mining

Given an access control system + Supporting data → Another access control model

Mining is partially automated so far…

Problem Statement

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

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

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

Contributions

- Feasibility analysis on ReBAC policy mining for the <u>first time</u>
- Developing feasibility analysis algorithms for the given set of criteria with complexity analysis
  - Variety of ReBAC rule structures are considered
- In case of infeasibility, solution algorithms are presented to make it feasible under given criteria
  - Varieties available
- Demonstrate the generated algorithms with cases and show the effectiveness beyond complexity analysis
- Future scopes

ReBAC Rule Structure

$$Rule_{op} ::= Rule_{op} \vee Rule_{op} \mid pathRuleExpr$$
$$pathRuleExpr ::= pathRuleExpr \wedge pathRuleExpr \mid pathLabelExpr$$
$$pathLabelExpr ::= pathLabelExpr.pathLabelExpr \mid edgeLabel$$
$$edgeLabel ::= \sigma, \sigma \in \Sigma$$

- Evaluation of access request (a, b, op)
  - for each pathLabelExpr in $Rule_{op}$ substitute True if there exists a simple path p from a to b in RG with path label pathLabelExpr, otherwise substitute False
  - the resulting boolean expression evalutes true → grant, deny otherwise

Feasibility Detection

Input:
Authorizations
RG
ReBAC rule structure

RG is directed

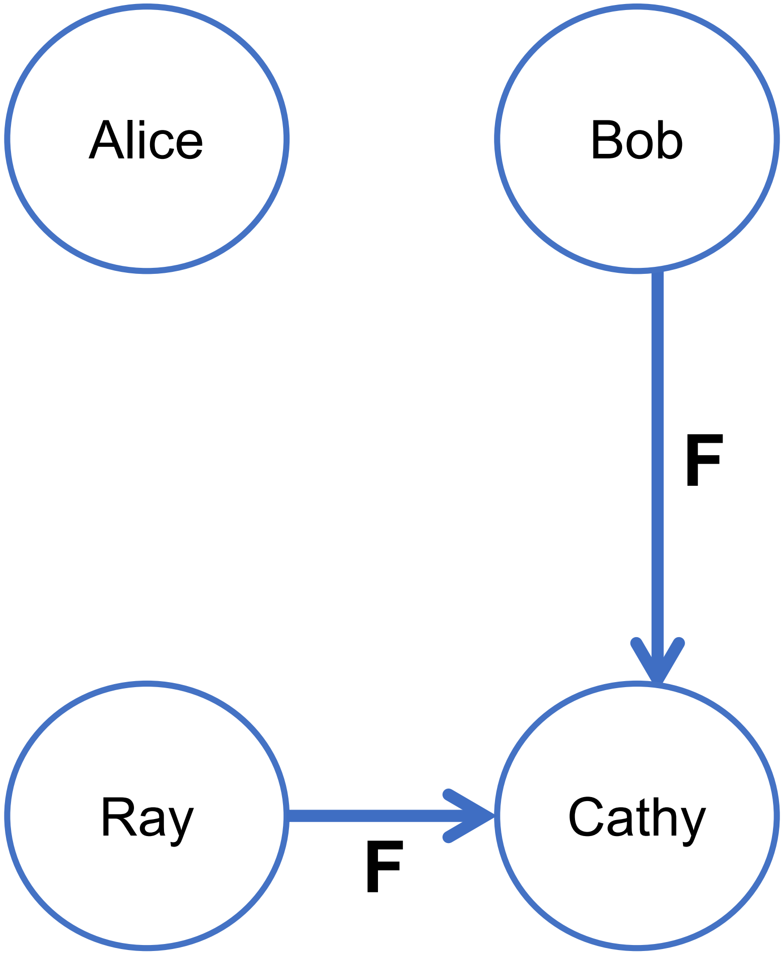
Feasibility detection Algorithm

Complexity !!

Output:
Feasible / Infeasible Status

Failed authorization list is returned

RG Example



Feasible

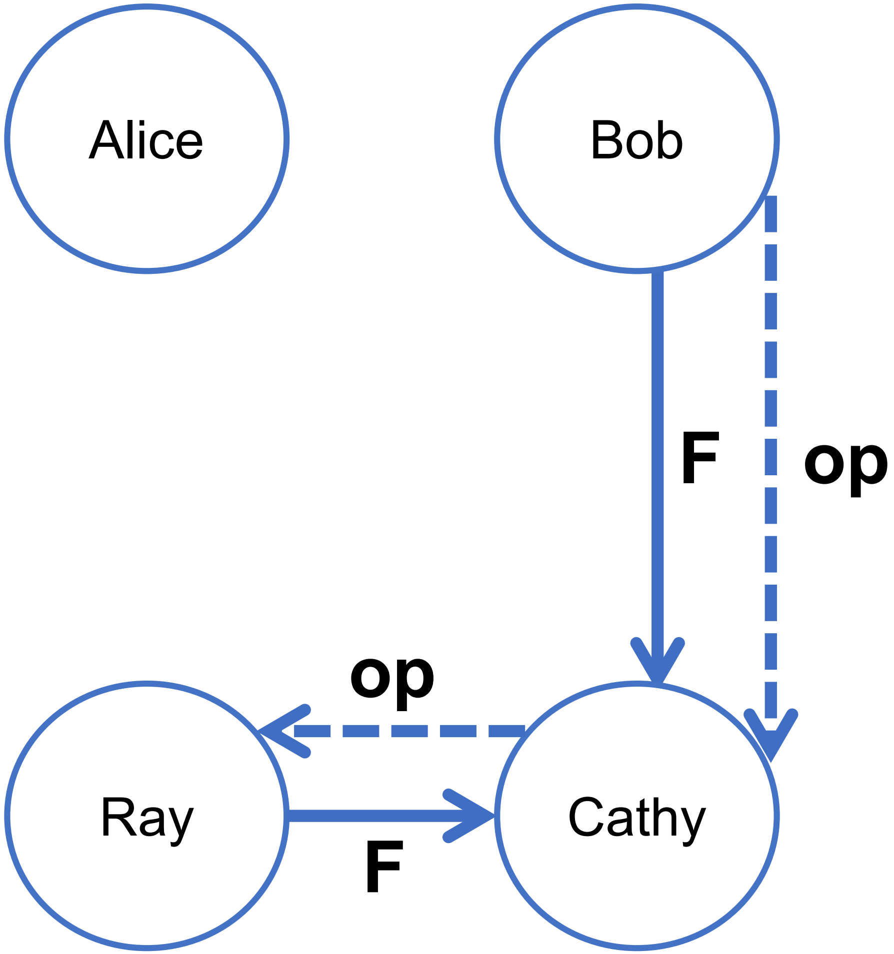
(Bob, Cathy, op)
(Ray, Cathy, op)

$Rule_{op}$ = F

Infeasible

i) (Bob, Cathy, op)
ii) (Cathy, Ray, op)

Solution 1

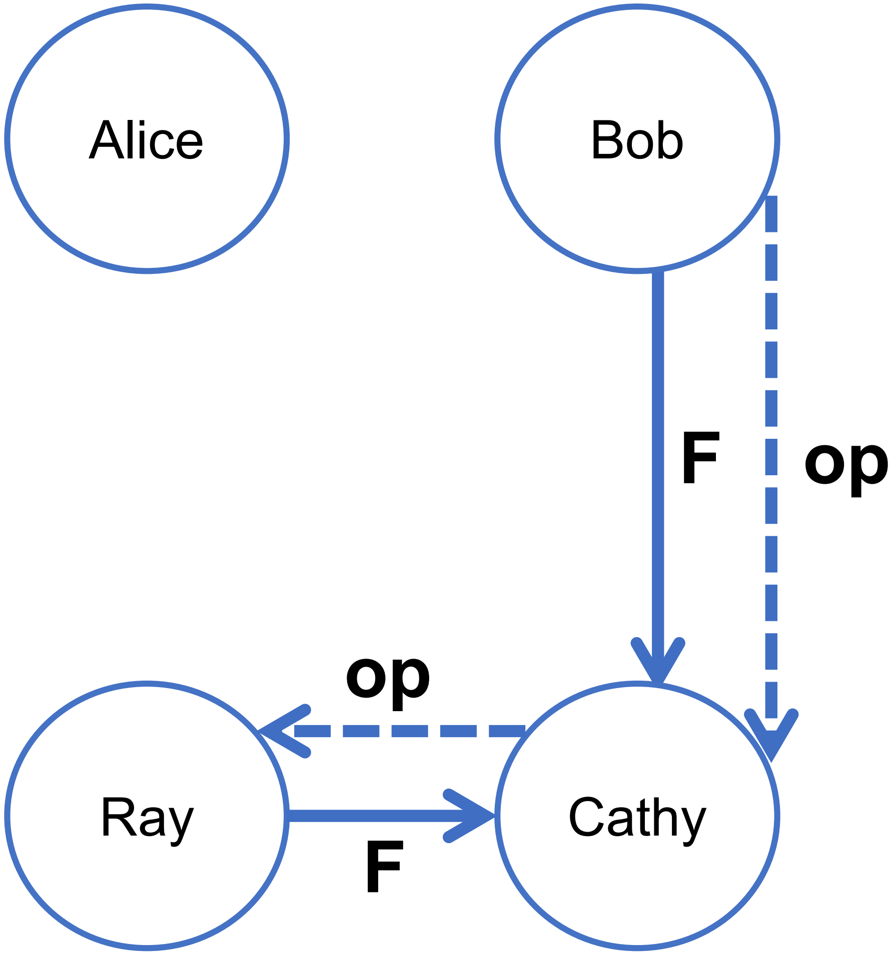


Infeasible

i) (Bob, Cathy, op)
ii) (Cathy, Ray, op)

$Rule_{op}$ = op

Solution 1



Infeasible

i) (Bob, Cathy, op)
ii) (Cathy, Ray, op)

Rule_{op} = op

Simple

Operation ∩ Relationship types={}

Minimal edges not guaranteed

|Authorization| edges at worst!

Path Variations

Simple Path (SP)

Simple Complementary Path (SCP)

Simple Permissive Path (SPP)

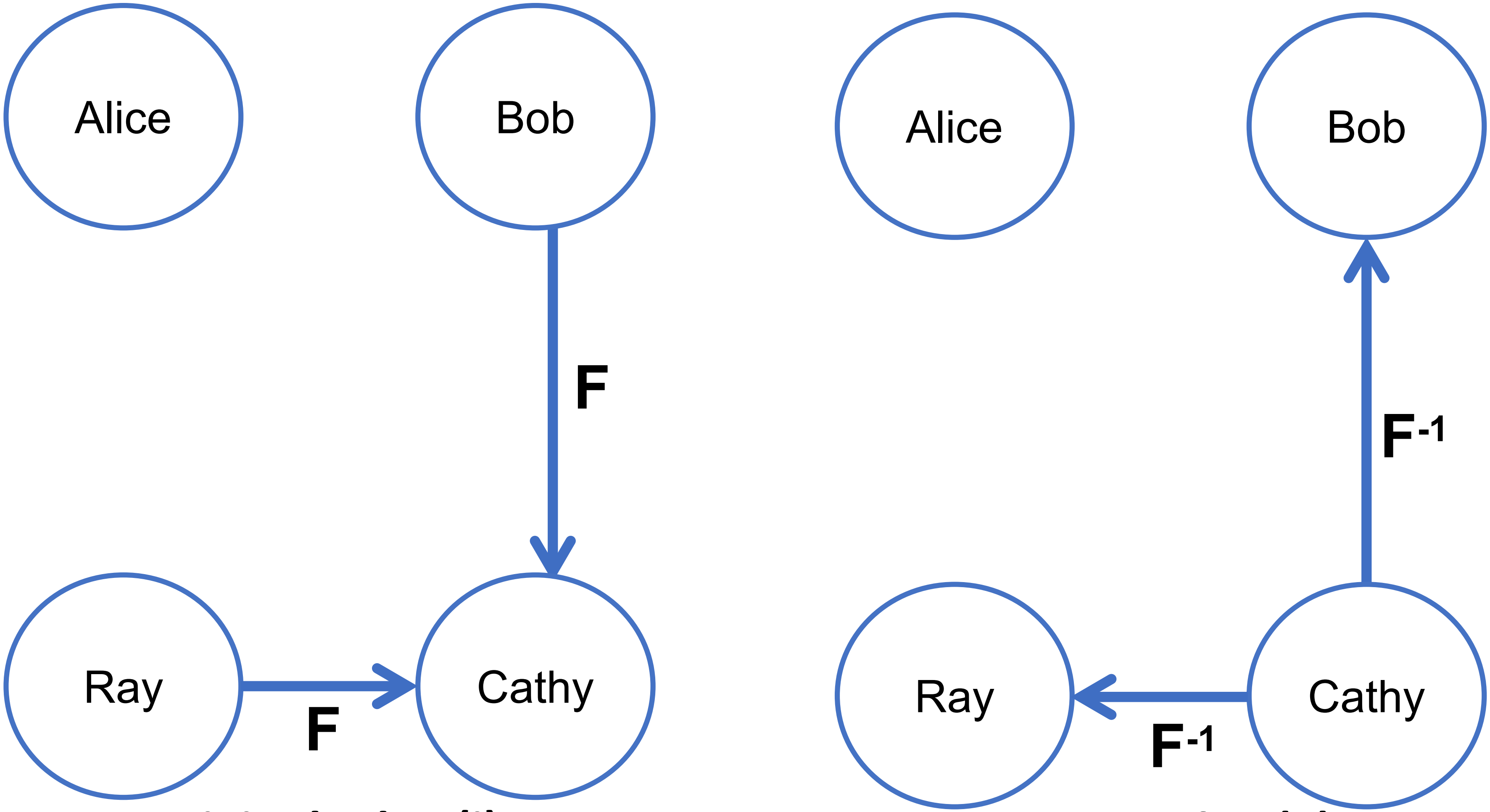
Simple Complementary Permissive Path (SCPP)

Characteristics	SCP	SPP	SCPP
$(a, b, \sigma) \rightarrow (a, b, \sigma) \in E, \sigma \in \Sigma$	✓	✓	✓
$(a, b, \overline{\sigma}) \rightarrow (a, b, \sigma) \notin E, \overline{\sigma} \in \overline{\Sigma}$	✓		✓
$(a, b, \sigma^{-1}) \rightarrow (b, a, \sigma) \in E, \sigma^{-1} \in \Sigma^{-1}$		✓	✓
$(a, b, \overline{\sigma}^{-1}) \rightarrow (b, a, \sigma) \notin E, \overline{\sigma}^{-1} \in \overline{\Sigma}^{-1}$			✓

Table represents path variations with original, non-relationship, inverse and non-relationship inverse edges (row 1, 2, 3, and 4, respectively).

- a,b: users, E and ∑ are the sets of edges and relationship type specifiers

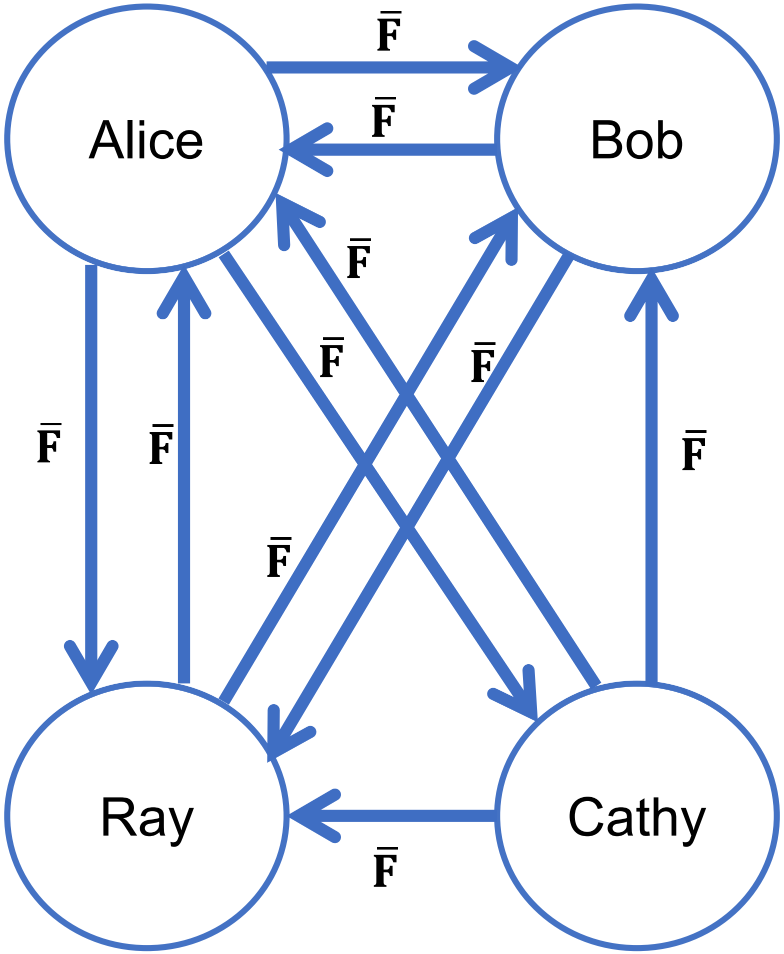
Path Variations Cont.



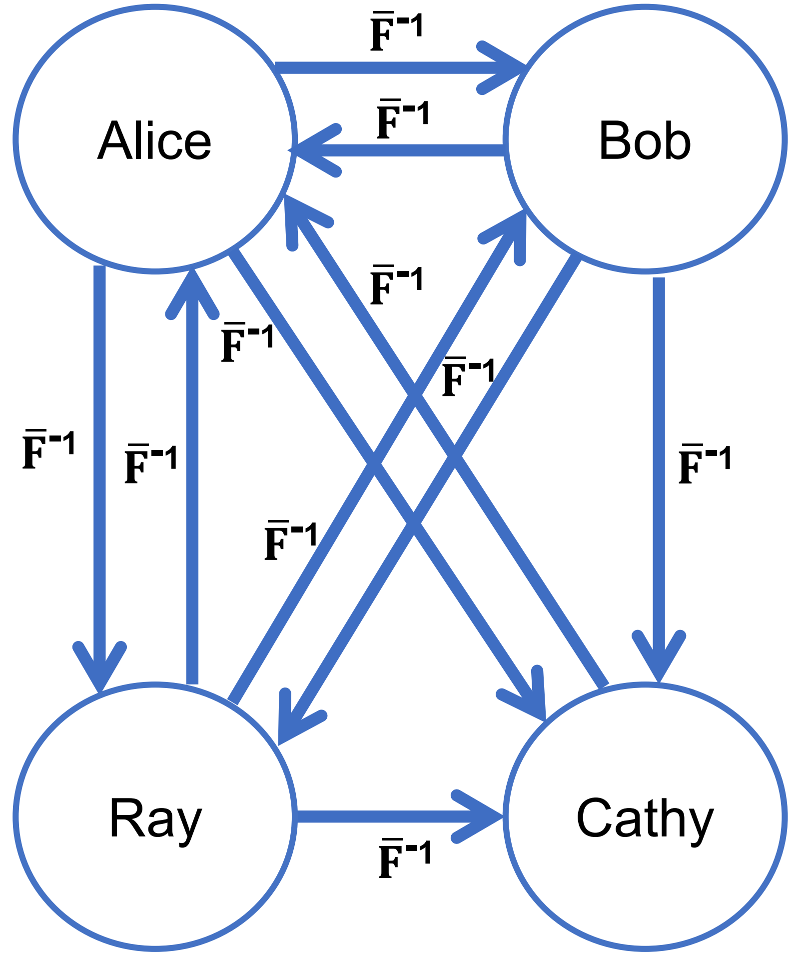
Original edge (i)

Inverse edge (ii)

Path Variations Cont.



Non-relationship edge (iii)



Non-relationship inverse edge (iv)

RREP Variations

RREP-0 → SP (i)

RREP-1 → SCP (i + iii)

RREP-2 → SPP (i + ii)

RREP-3 → SCPP (i + ii + iii + iv)

Rule minimization techniques are described in the paper

Future Enhancement

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

!! Just the beginning !!

Acknowledgement

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

❖ Question/ Feedback