Efficient Algorithms for Association Finding and Frequent Association Pattern Mining

Gong Cheng, Daxin Liu, Yuzhong Qu

Websoft Research Group
National Key Laboratory for Novel Software Technology
Nanjing University, China

Websoft

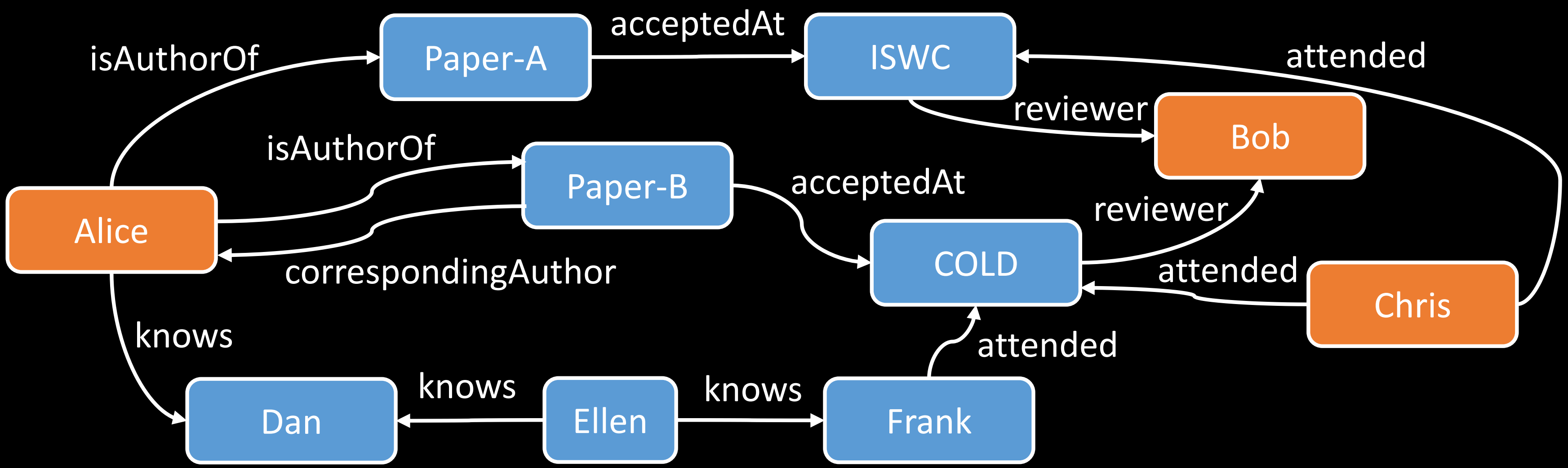
Background and motivation

- To suggest friends, recognize suspected terrorists, answer questions … based on massive graph data

Problem statement

An *association* connecting a set of query entities is a minimal subgraph that

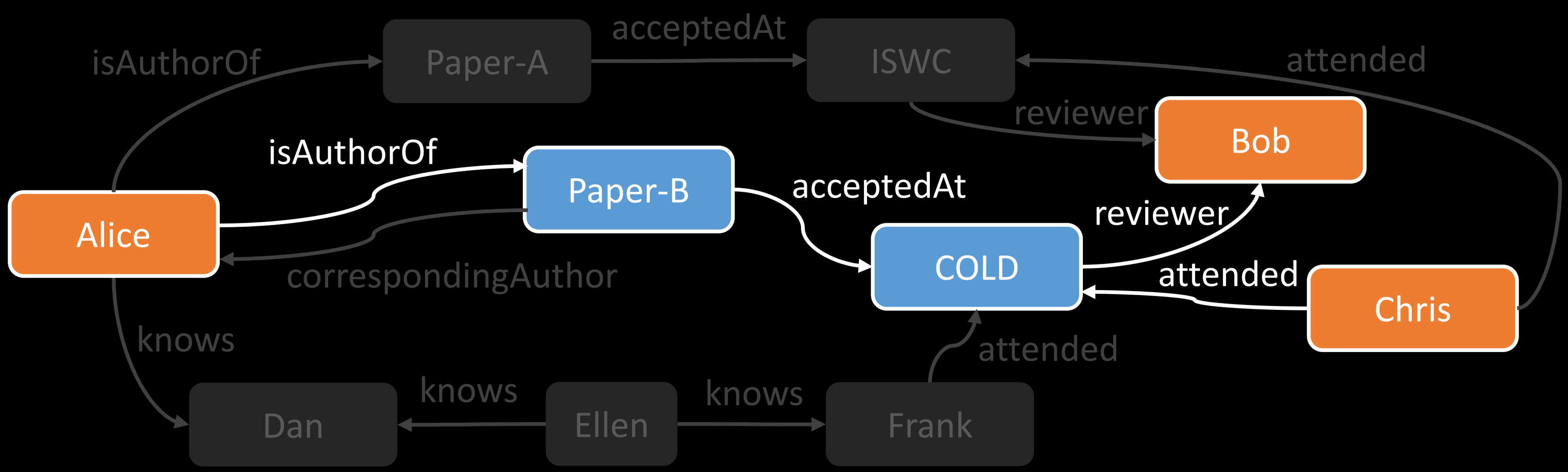
- contains all the query entities, and
- is connected.



Problem statement

An *association* connecting a set of query entities is a minimal subgraph that

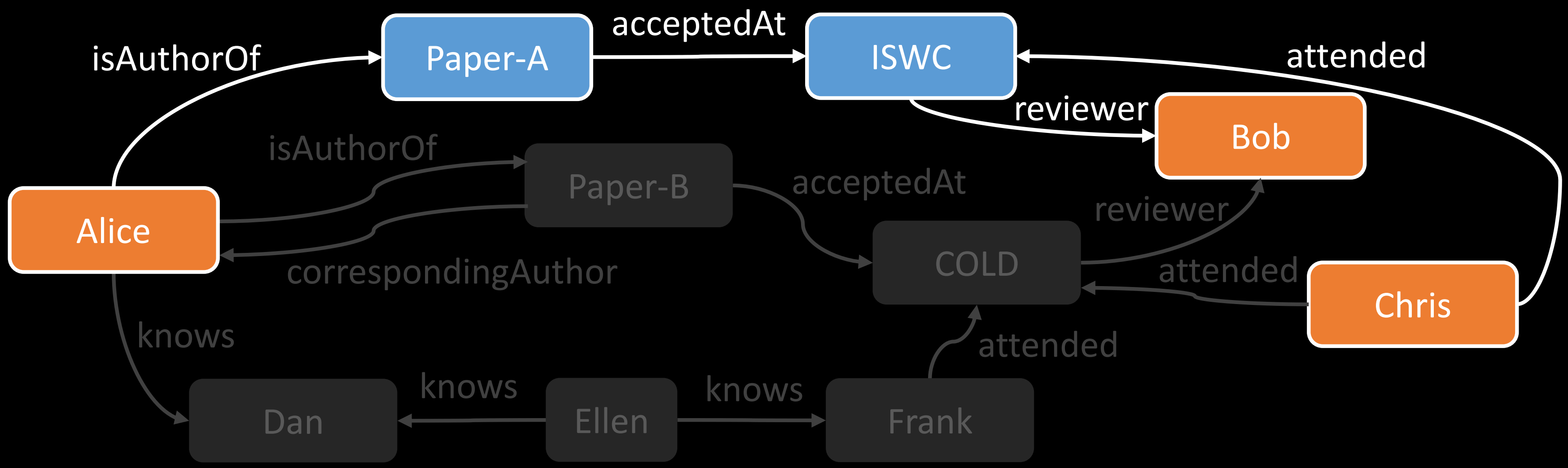
- contains all the query entities, and
- is connected.



Problem statement

An *association* connecting a set of query entities is a minimal subgraph that

- contains all the query entities, and
- is connected.

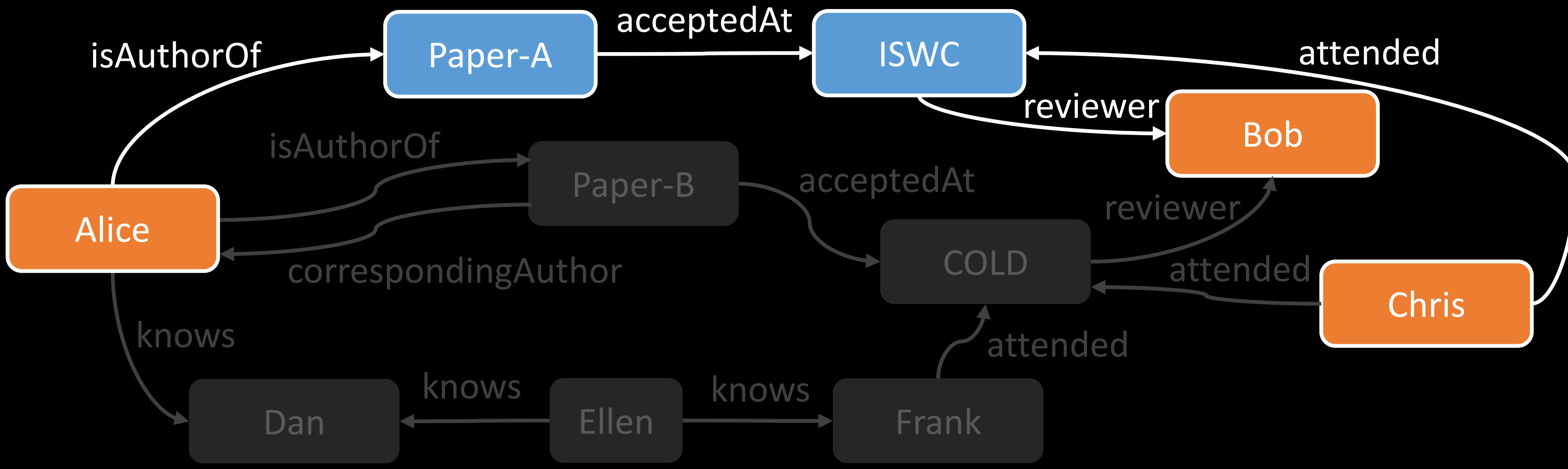


Problem statement

An *association* connecting a set of query entities is a <u>minimal</u> subgraph that

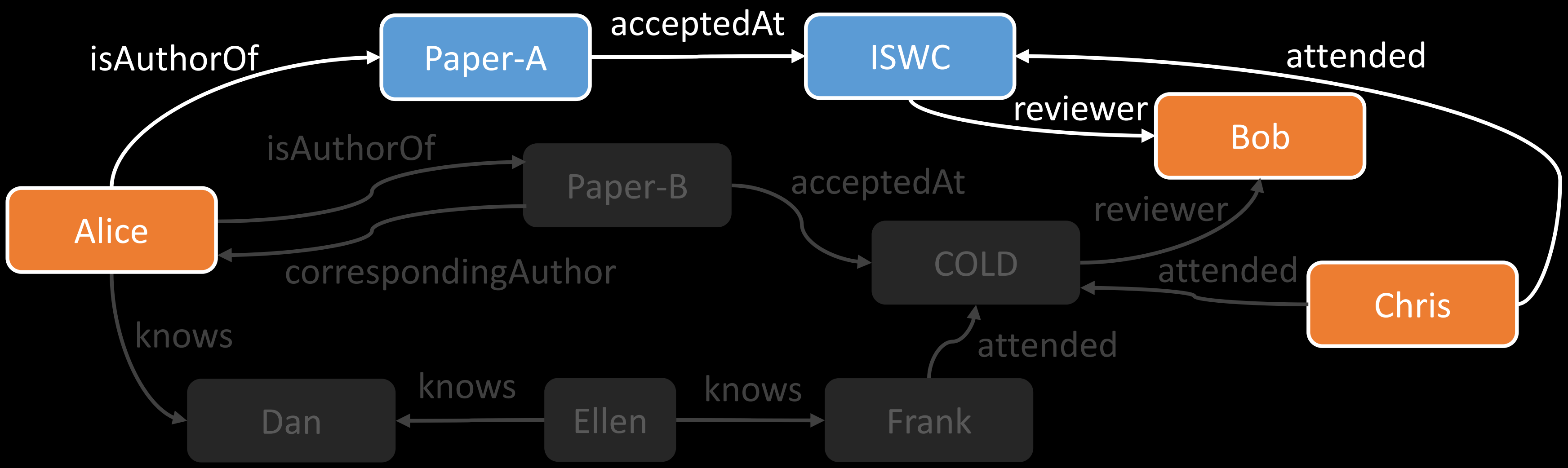
- contains all the query entities, and
- is connected.

- **Tree-structured**
- **Leaves ⊆ Query entities**



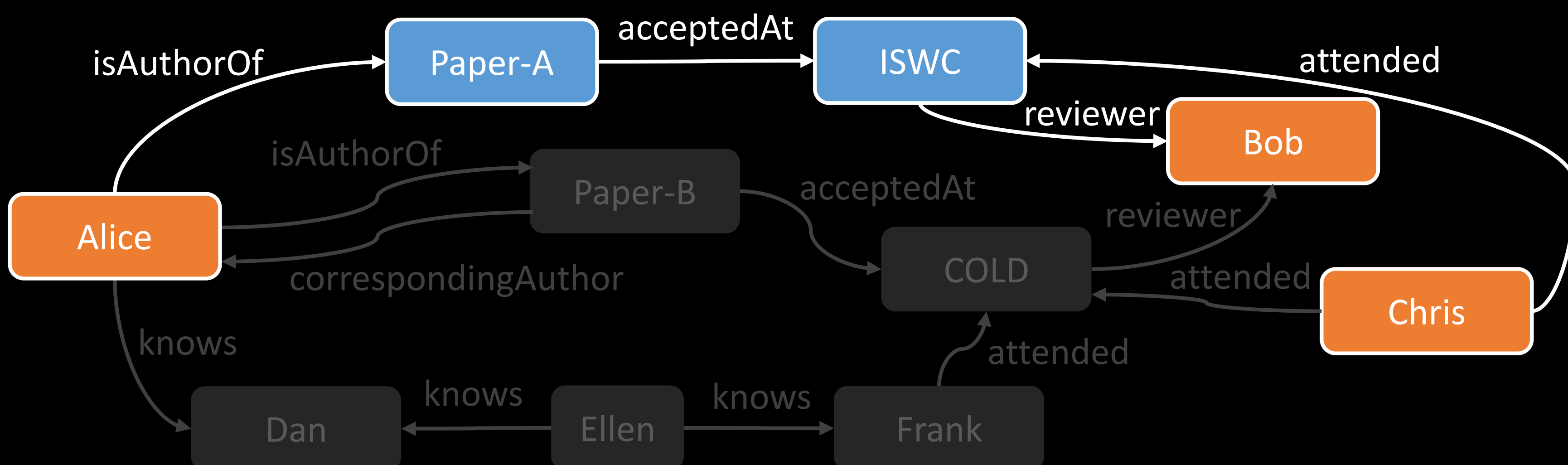
Problem statement

1. How to efficiently find associations in a possibly very large graph?
2. How to help users explore a possibly large set of associations that have been found?



Problem statement

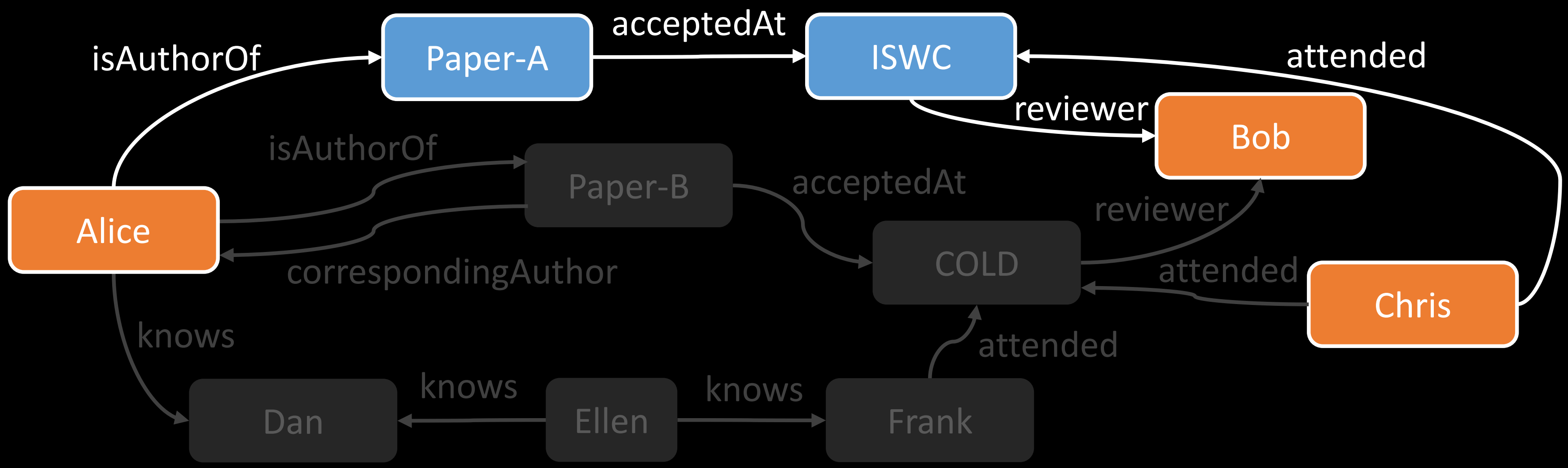
1. How to efficiently find associations in a possibly very large graph? Association finding
2. How to help users explore a possibly large set of associations that have been found? Frequent association pattern mining



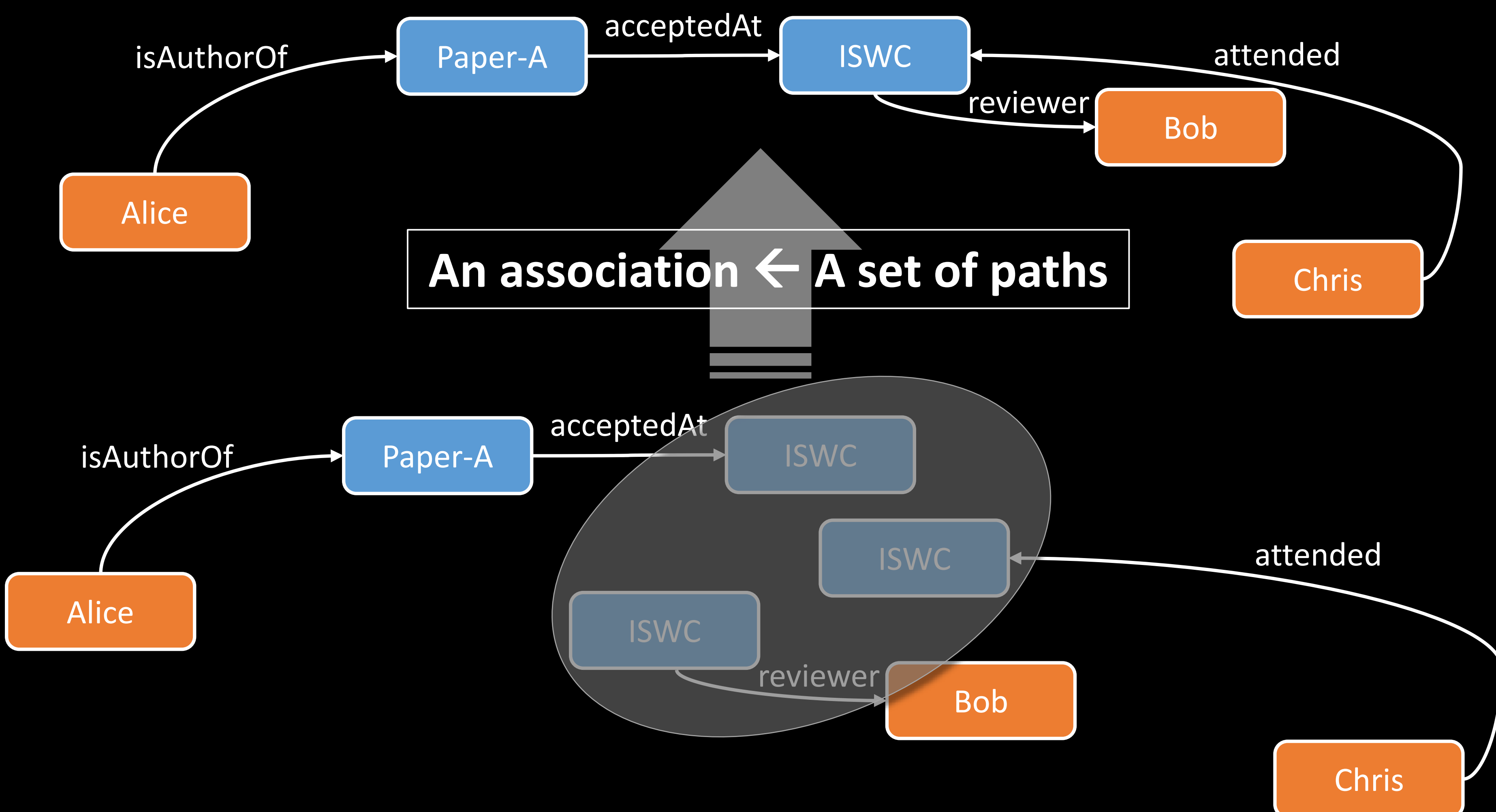
Association finding: Problem

- To find all the associations having a limited diameter

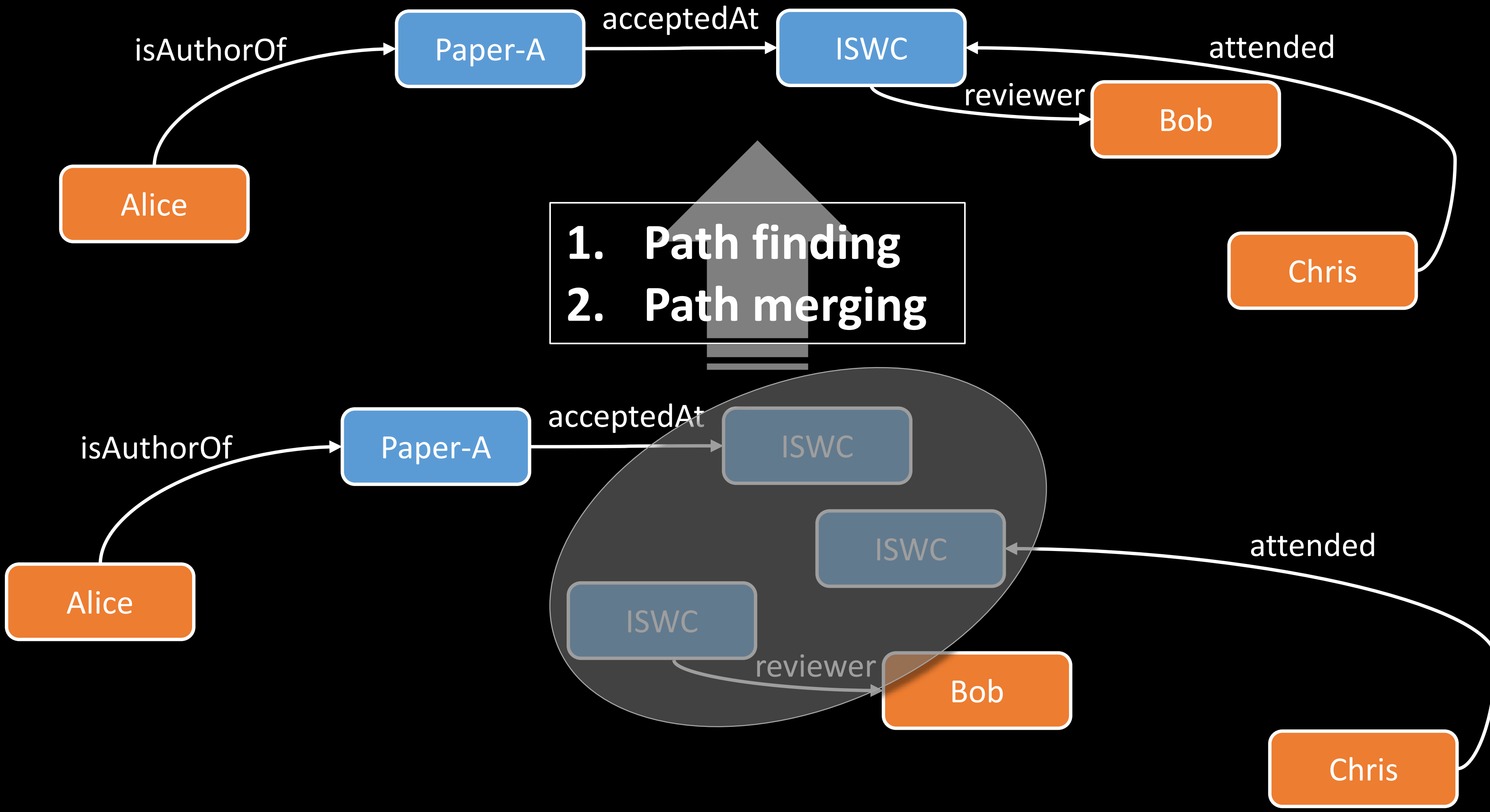
  (Diameter = Greatest distance between any pair of vertices)



Association finding: Basic solution

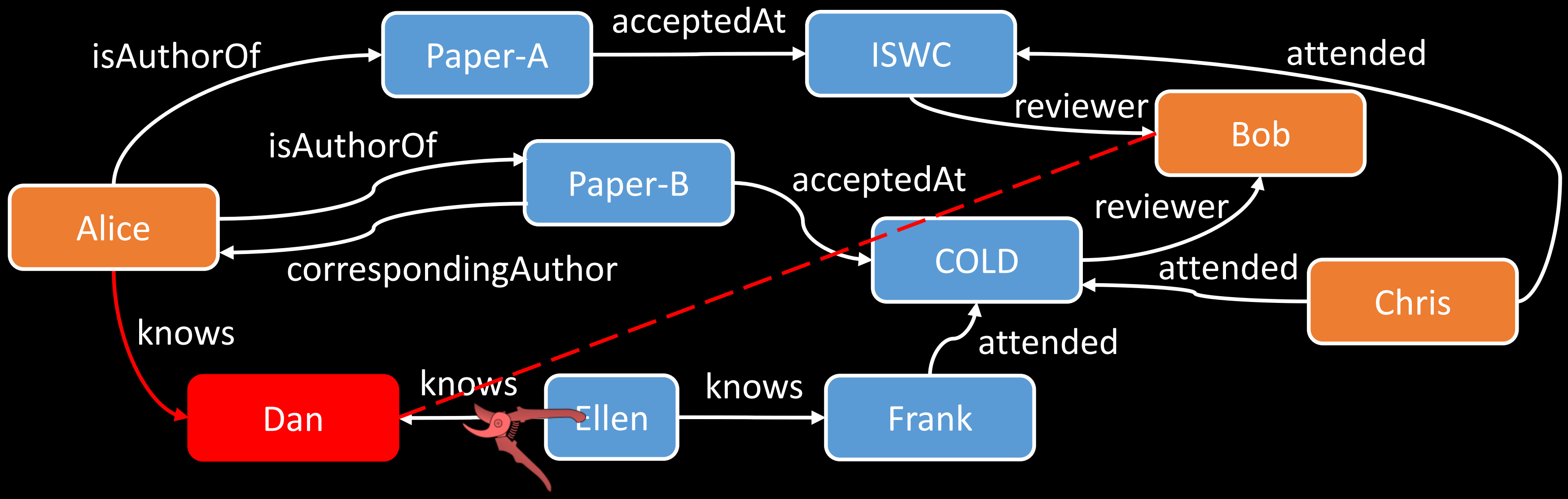


Association finding: Basic solution



Association finding: Optimization

- Distance-based search space pruning

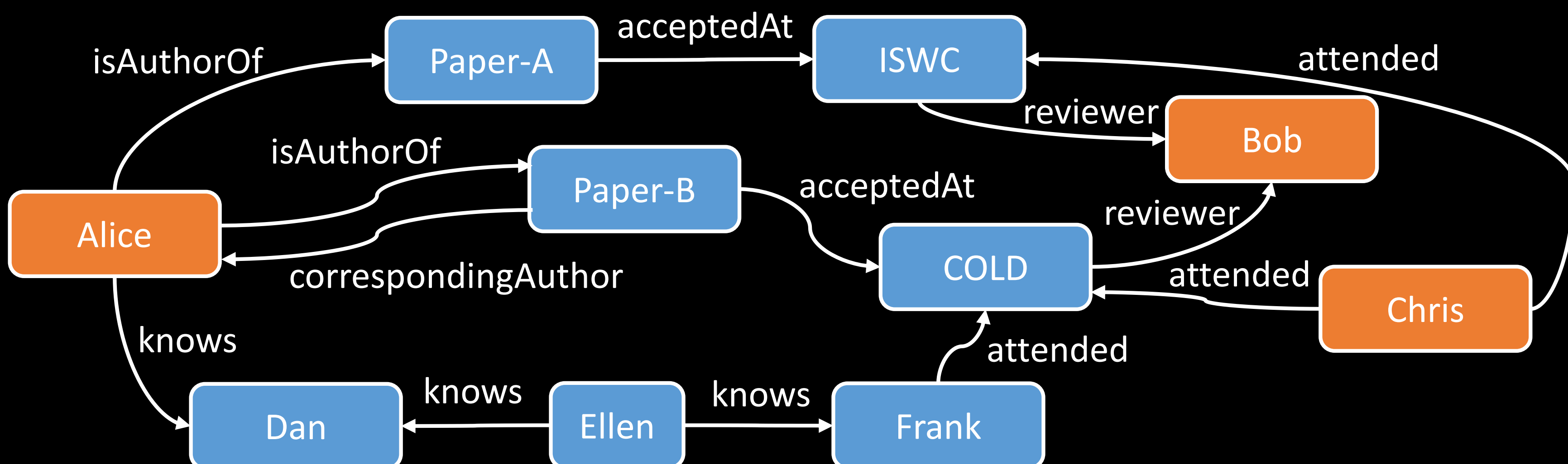


- DiameterConstraint ≤ 3
- Length(Alice→Dan) = 1
- Distance(Dan, Bob) = 4

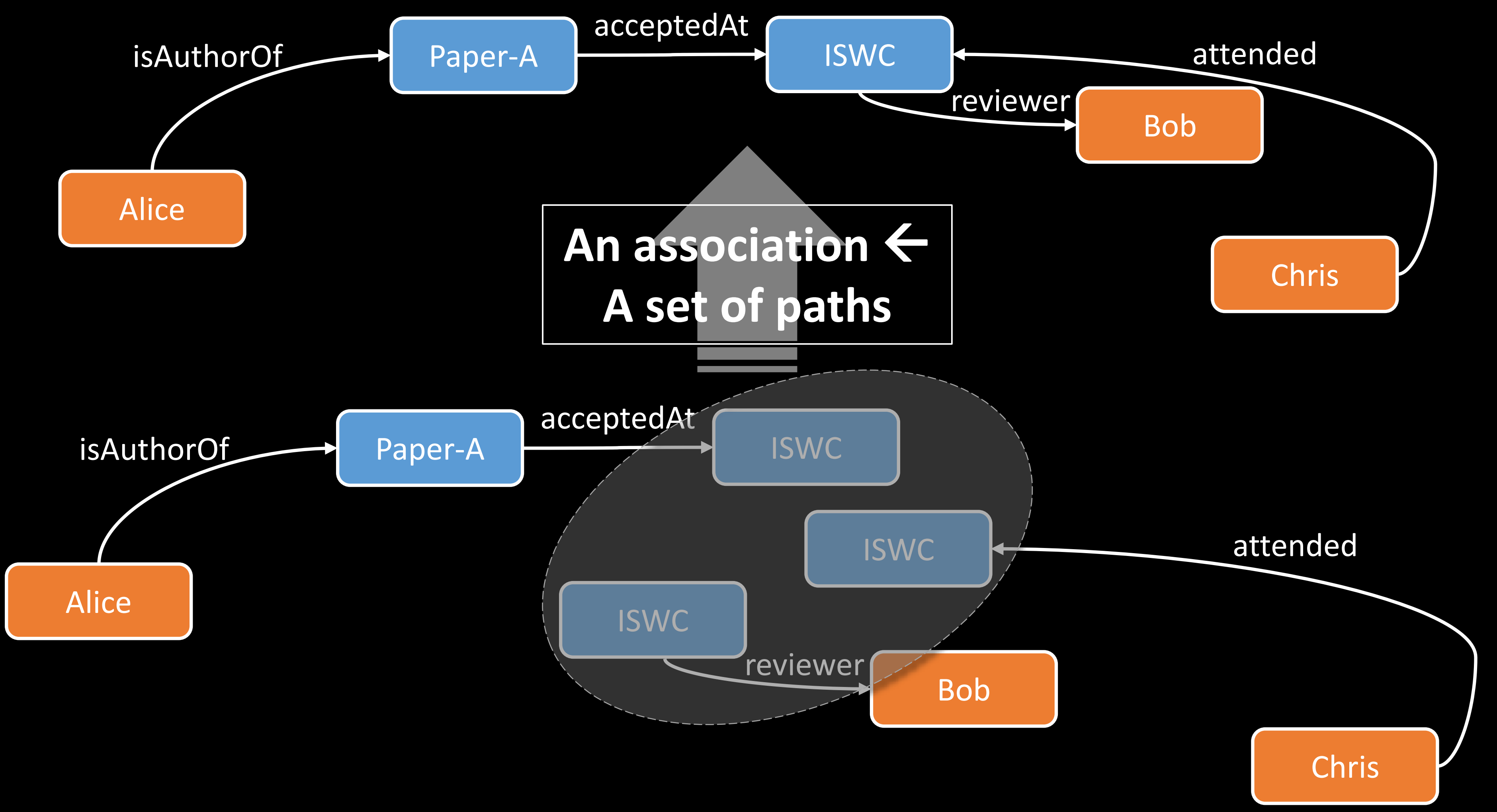
Length+Distance > DiameterConstraint

Association finding: Optimization

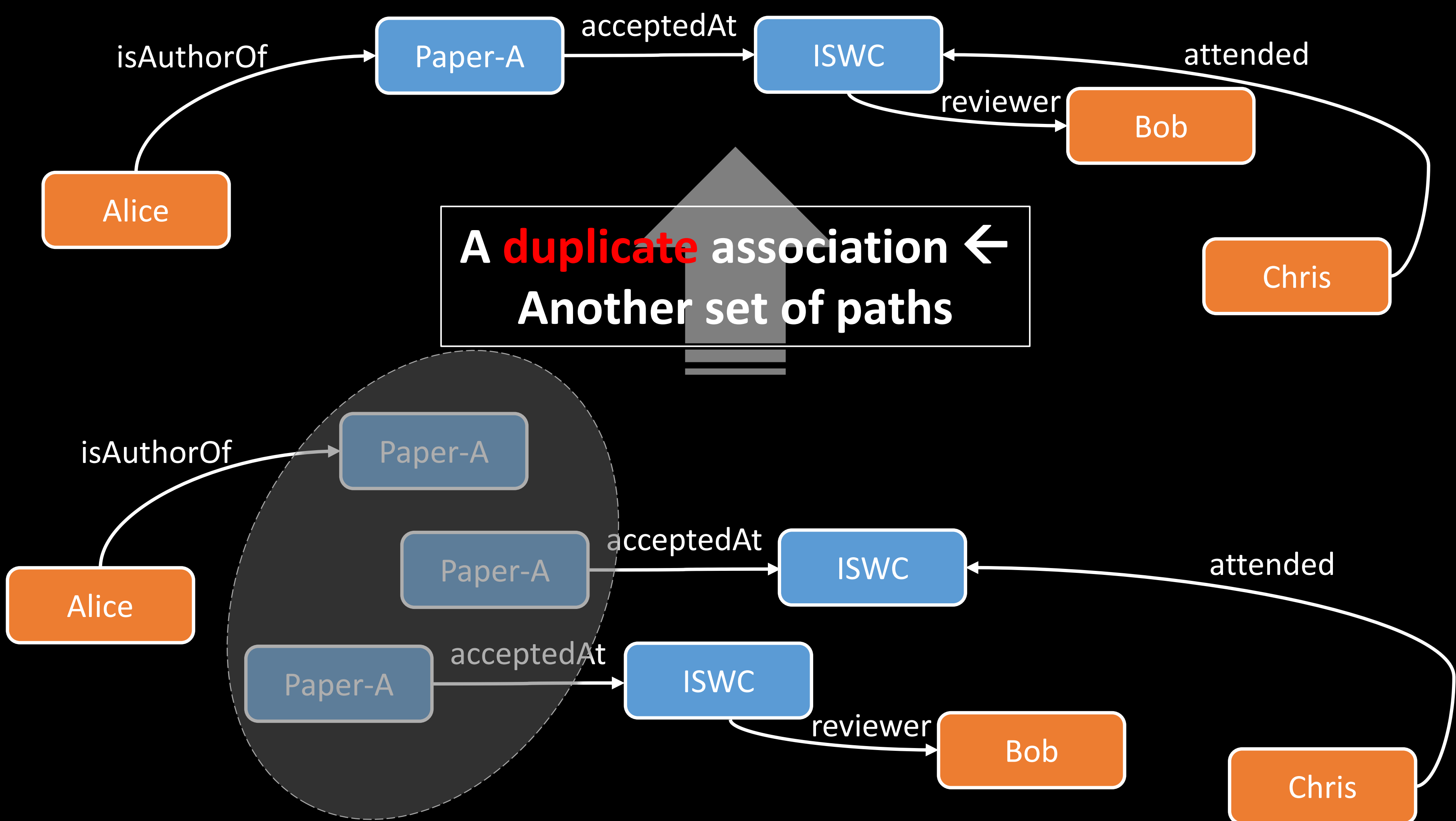
- Distance computation
  - Materializing offline computed results: $O(V^2)$ space
  - Online computing: $O(E)$ time per pair
  - Using distance oracle: a space-time trade-off



Association finding: Deduplication



Association finding: Deduplication

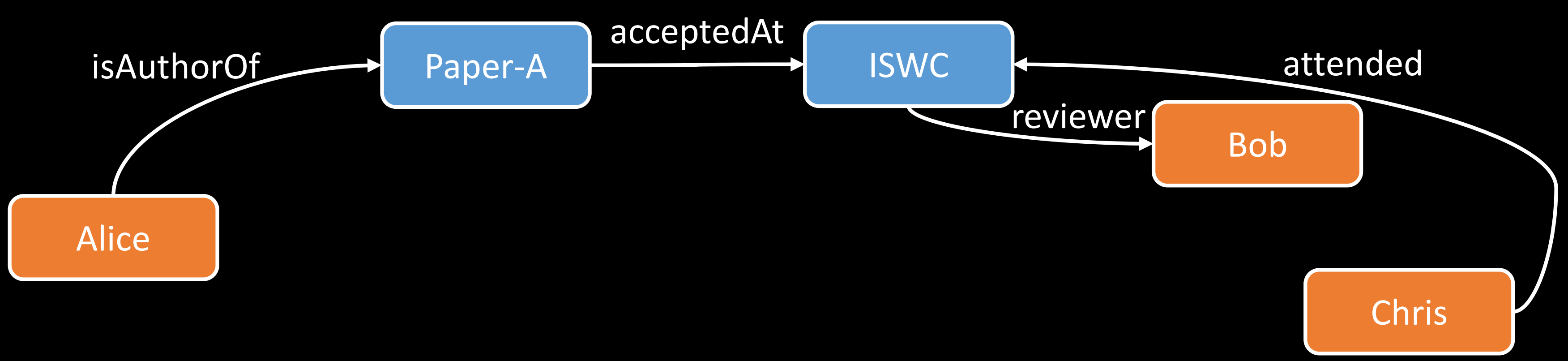


Association finding: Deduplication

- Canonical code

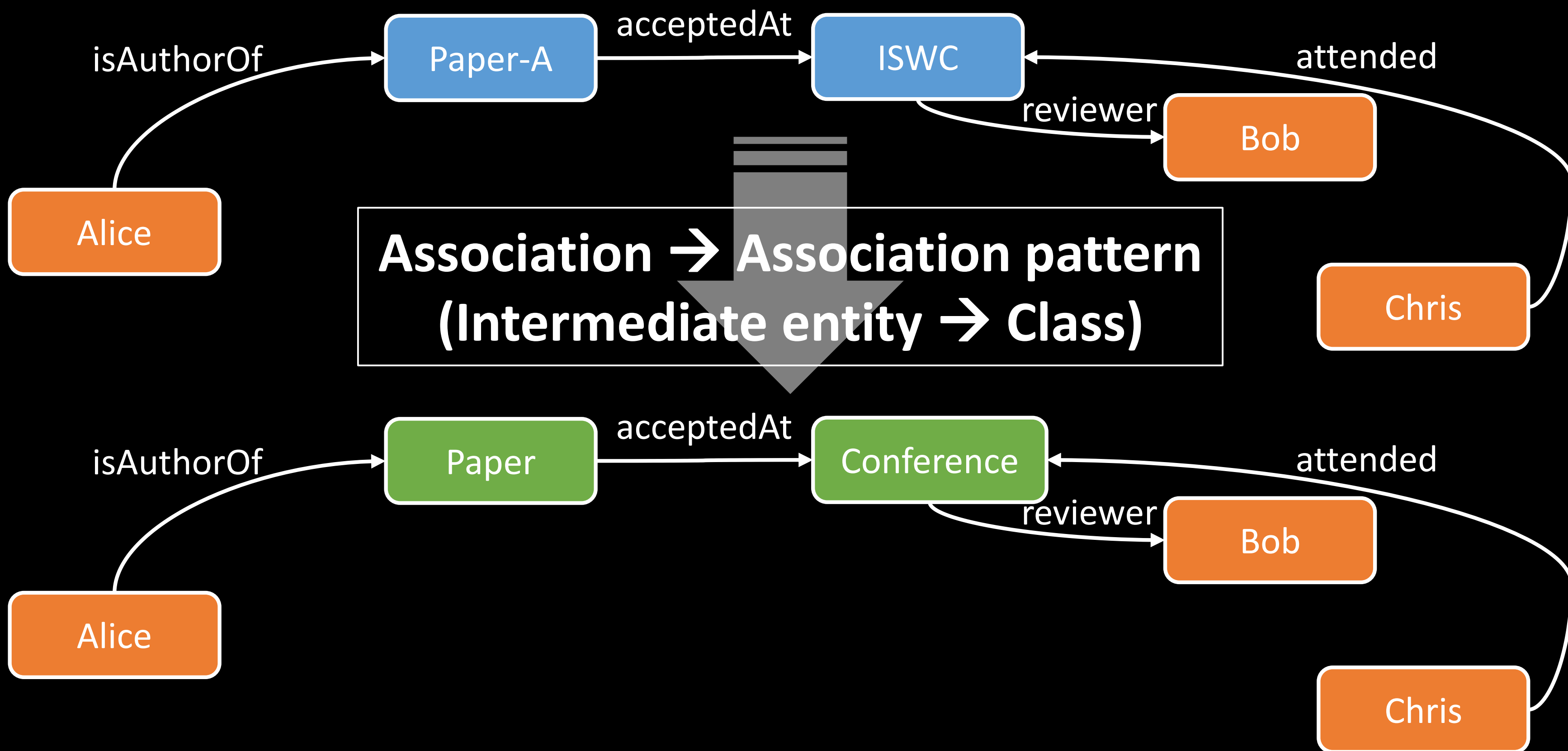
code(Tree(*Alice*)) = *Alice*,*isAuthorOf*,code(Tree(*Paper-A*))\$
= … *Paper-A*,*acceptedAt*,code(Tree(ISWC))\$ …
= … *ISWC*,*reviewer*,code(Tree(*Bob*)),~*attended*,code(Tree(*Chris*))\$ …
= … *Bob*\$ …

(Assuming *Bob* precedes *Chris*)



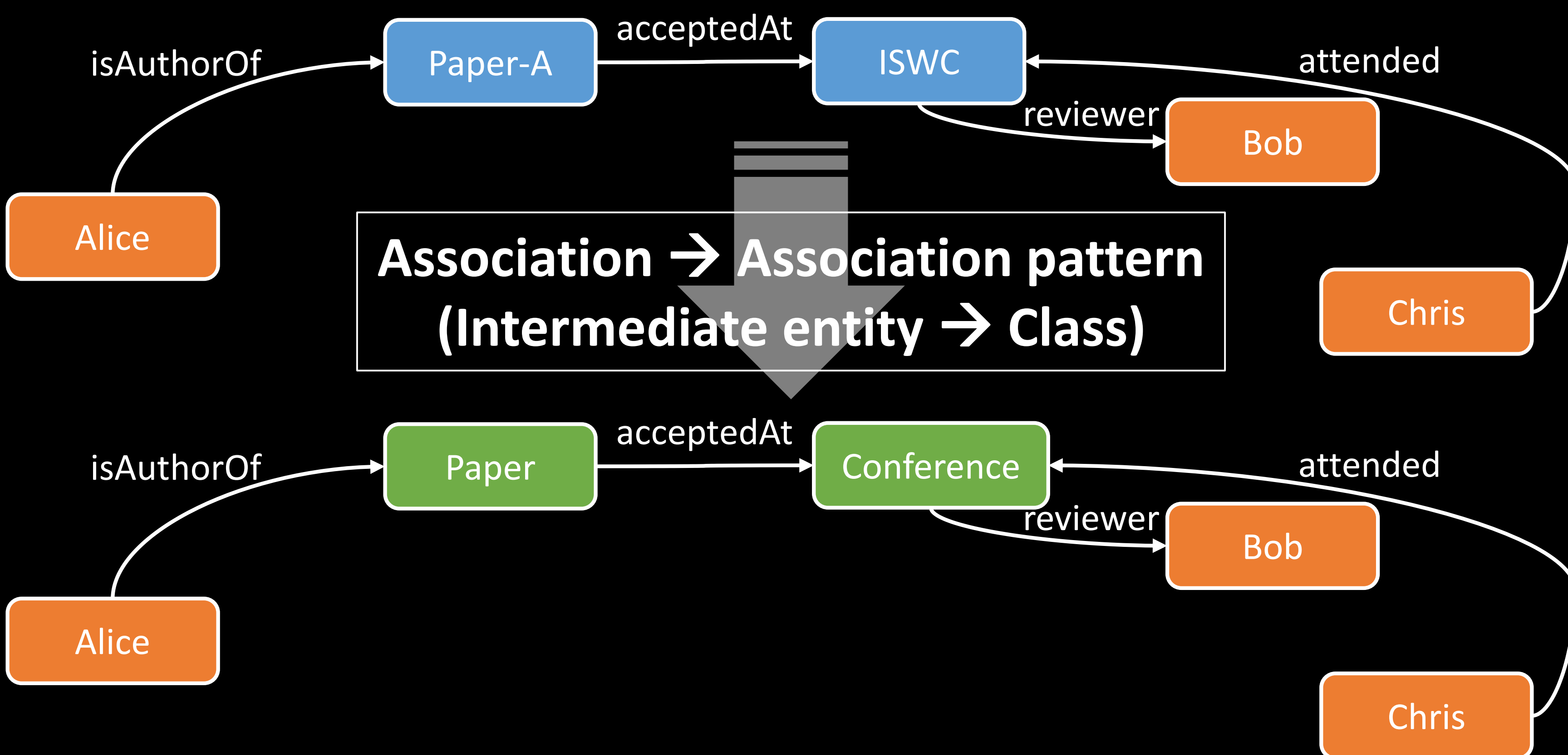
Frequent association pattern mining

- Association pattern: A conceptual abstract that summarizes a group of associations



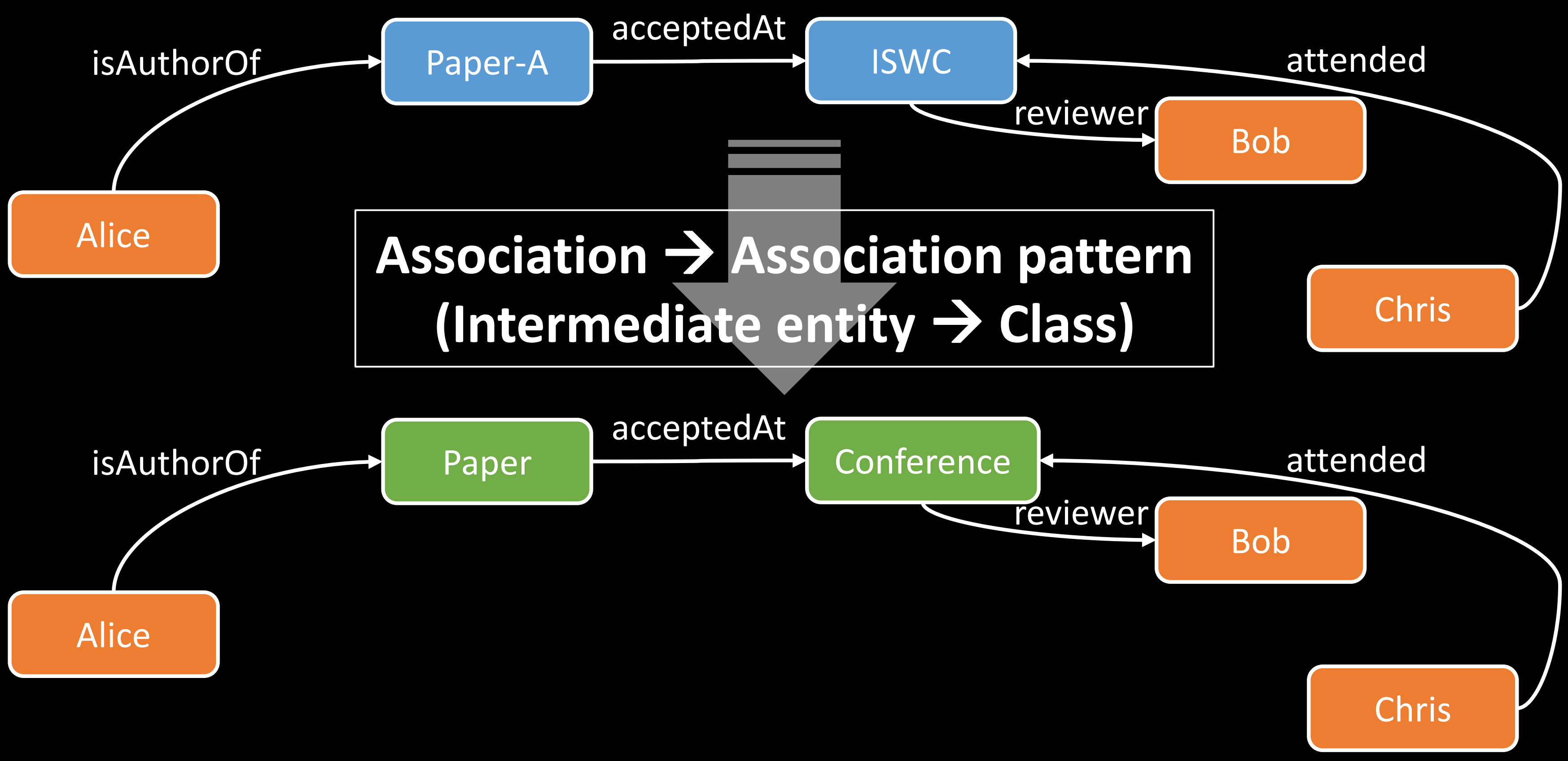
Frequent association pattern mining

- Problem: To mine all the association patterns matched by more than a threshold proportion of associations



Frequent association pattern mining

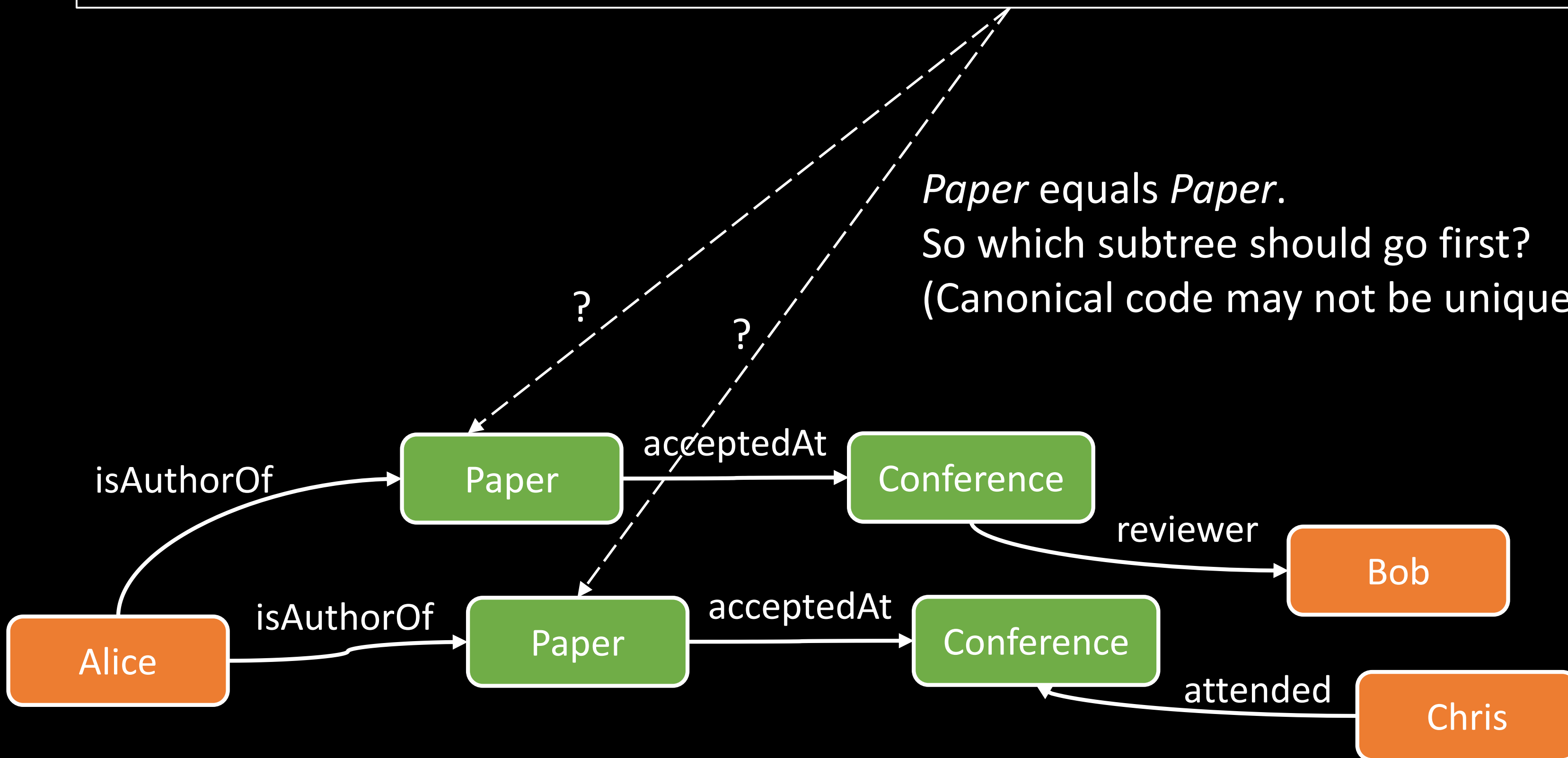
- Basic solution: Calculating the frequency of an association pattern
  = Counting the occurrence of its canonical code



Frequent association pattern mining

- Canonical code

code(Tree(*Alice*)) = *Alice*,*isAuthorOf*,code(Tree(*Paper*)),*isAuthorOf*,code(Tree(*Paper*))$



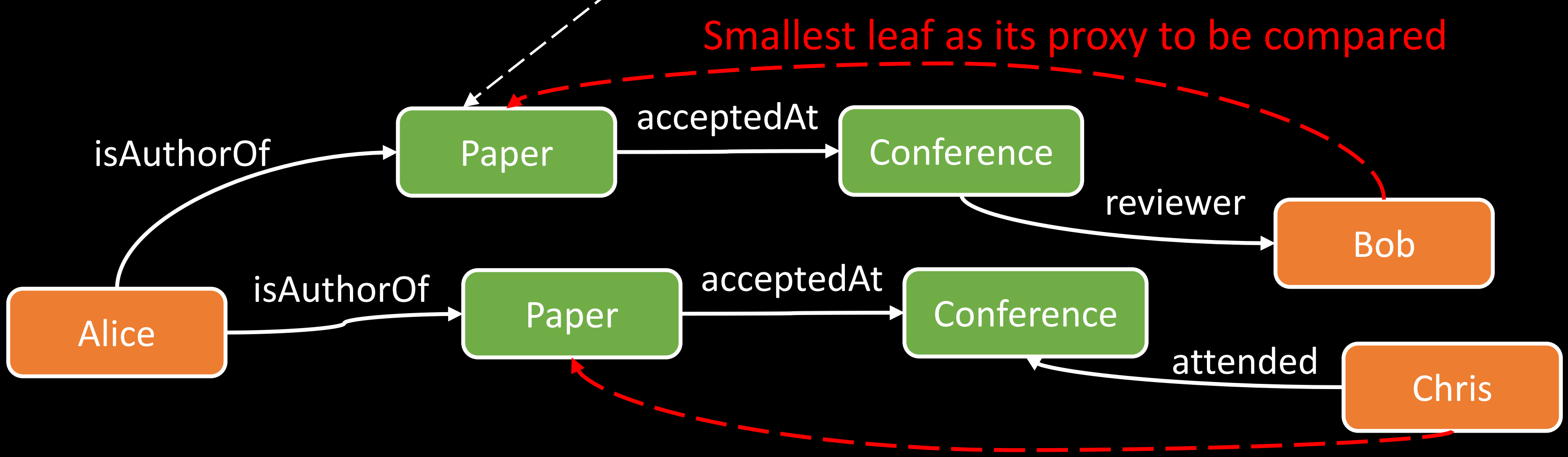
Frequent association pattern mining

- Canonical code

code(Tree(*Alice*)) = *Alice*,*isAuthorOf*,code(<u>Tree(*Paper*)</u>),*isAuthorOf*,code(<u>Tree(*Paper*)</u>)$

(Assuming *Bob* precedes *Chris*)

Equality would never happen.
(Canonical code is now unique!)



Experiments

- Datasets
  - LinkedMDB: 1M vertices and 2M arcs
  - DBpedia (2015-04): 4M vertices and 15M arcs
- Parameter settings
  - Diameter constraint (λ): 2, 4
  - Number of query entities (n): 2, 3, 4, 5
- Test queries
  - 1,000 random sets of query entities under each setting of λ and n
- Hardware configuration
  - 3.3GHz CPU, 24GB memory
  - Data graphs: in memory
  - Distance oracles: on disk

Experiments

- Results: Association finding

BSC: Basic solution (not pruning)
PRN: Optimized solution (distance-based pruning)
PRN-1: Optimized solution (distance-based pruning except for the last level of search)

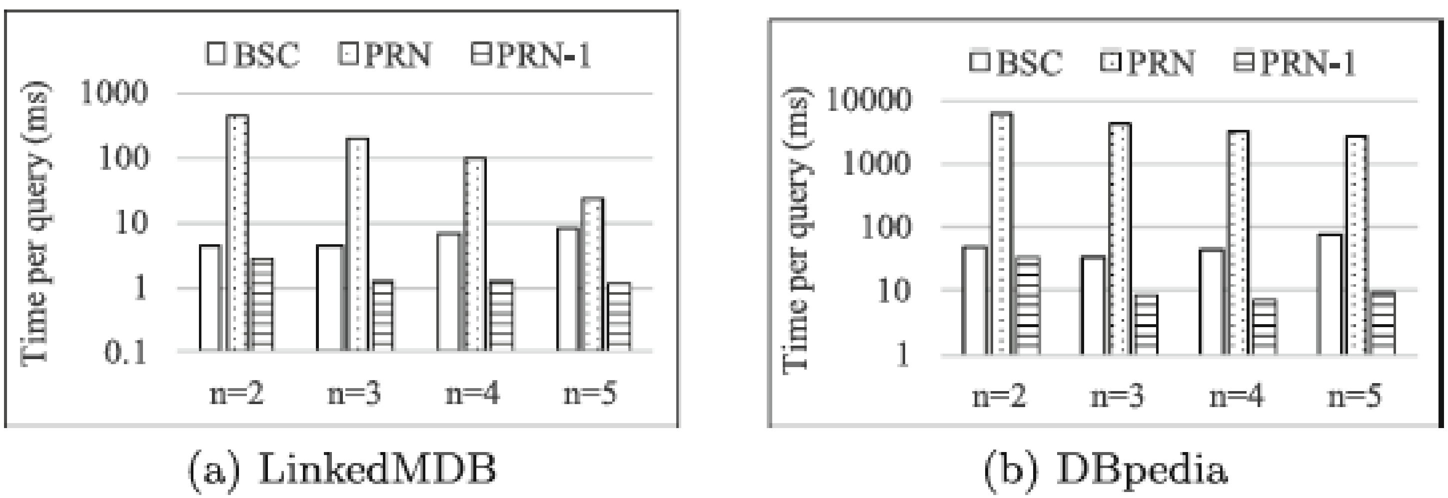


Fig. 5. Running time of association finding under $\lambda = 4$.

Experiments

- Results: Frequent association pattern mining
  - LinkedMDB
    - <10,000 associations: <21ms
    - 13,531 associations: 68ms
  - DBpedia
    - <10,000 associations: <65ms
    - 1,198,968 associations: 2909ms

Takeaway messages

- Subgraph finding and mining are faster than what we expected.
- Consider distance oracle and canonical code in your own research.

Takeaway messages

- Subgraph finding and mining are faster than what we expected.
- Consider distance oracle and canonical code in your own research.