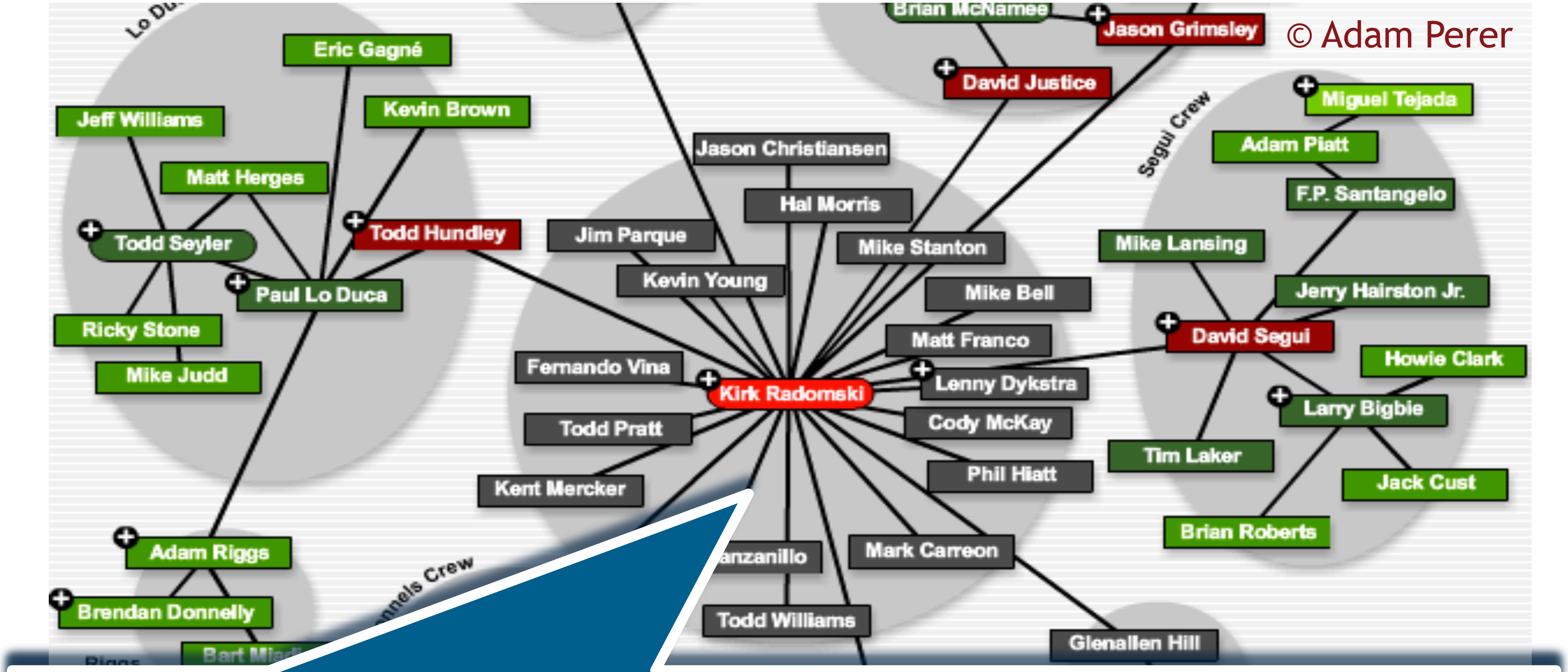


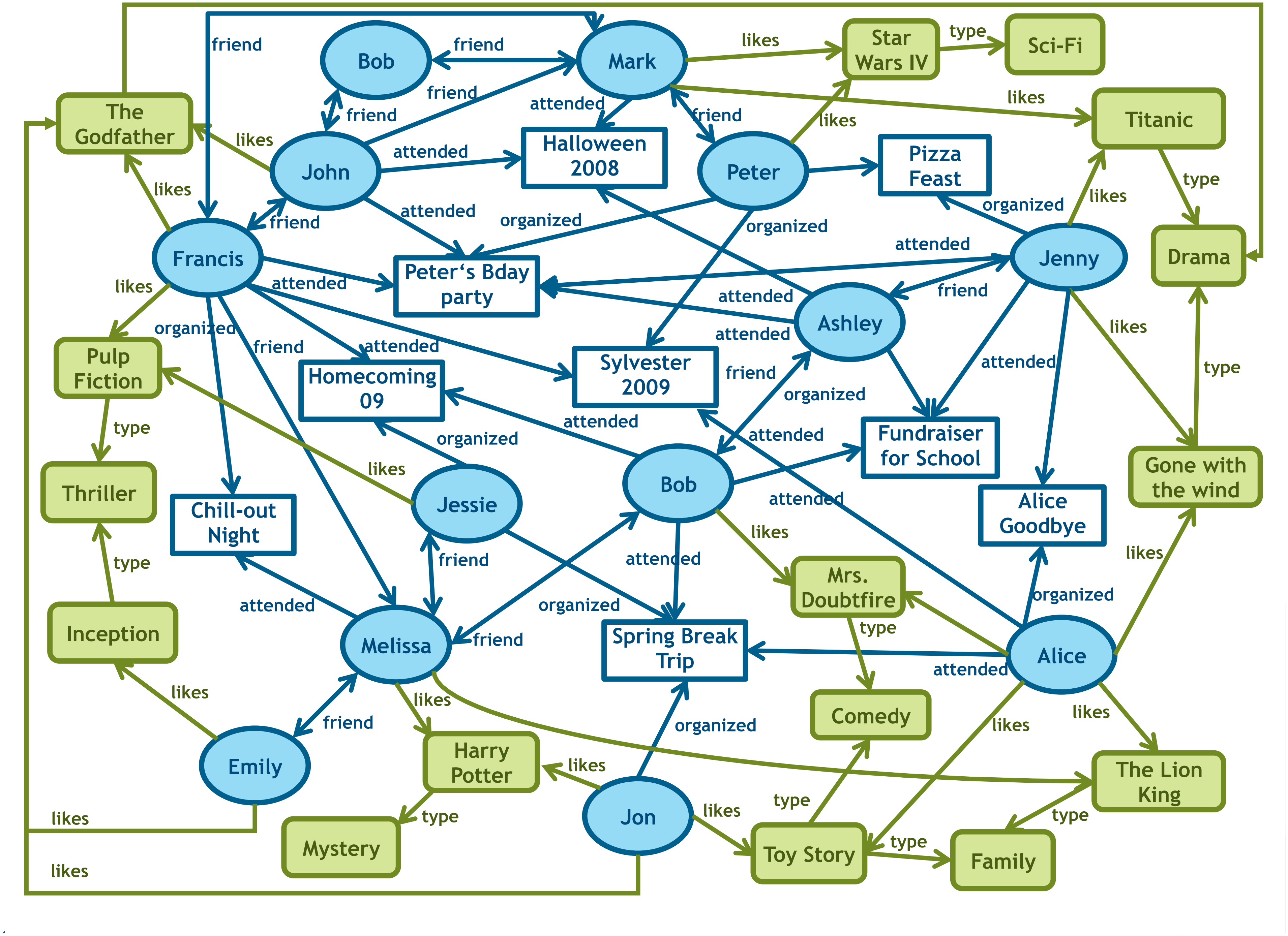
© Adam Perer

PMatch: Probabilistic Subgraph Matching on Huge Social Networks

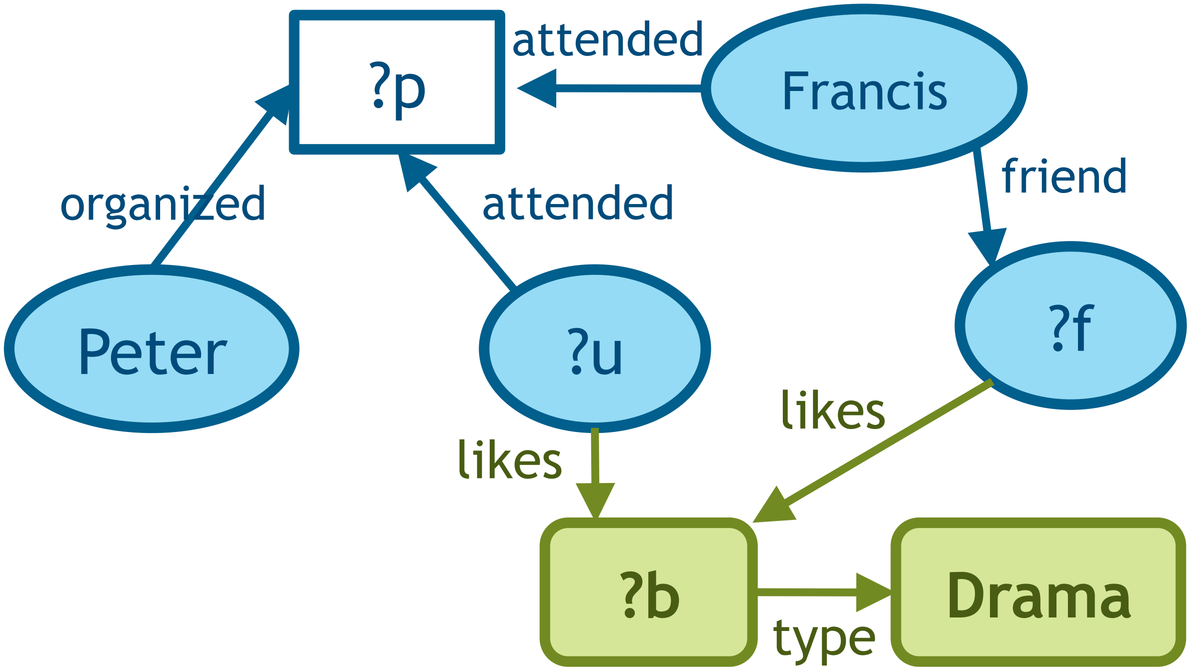
UNIVERSITY OF MARYLAND 18 56

Matthias Bröcheler, Andrea Pugliese
& V.S. Subrahmanian

UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

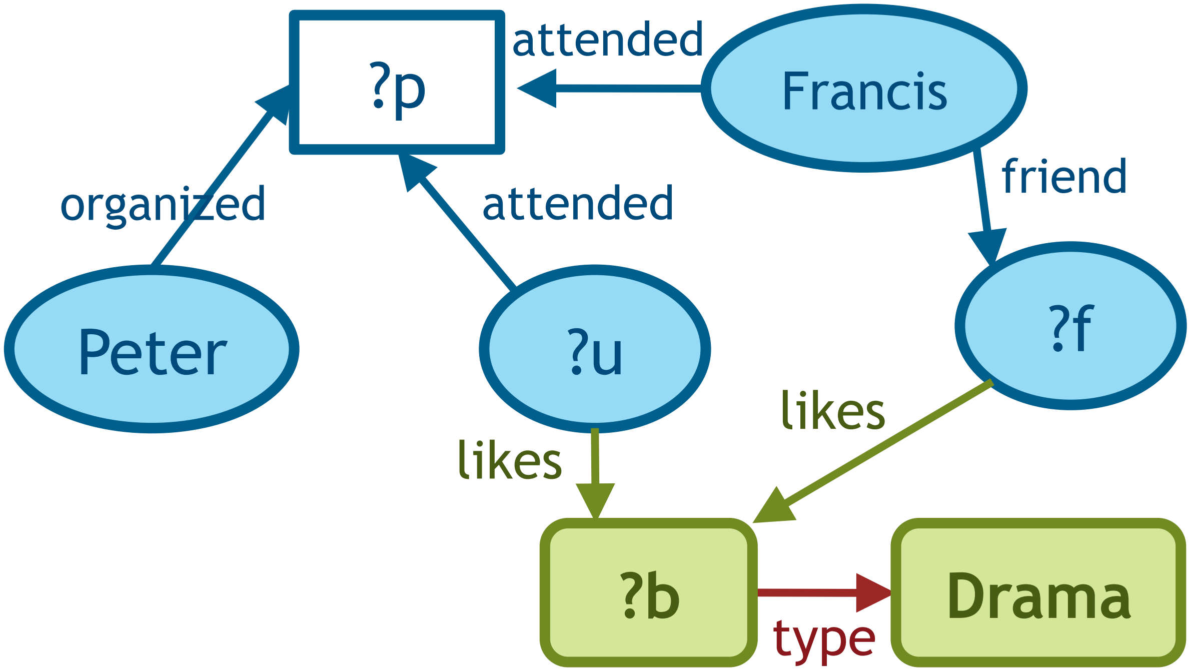

friend
Bob
friend
Mark
likes
Star Wars IV
type
Sci-Fi
The Godfather
friend
friend
attended
likes
Titanic
likes
friend
likes
Halloween 2008
John
attended
Peter
Pizza Feast
likes
type
organized
likes
attended
friend
organized
organized
Drama
Francis
Peter's Bday party
attended
Jenny
attended
friend
Ashley
likes
attended
attended
likes
organized
friend
attended
Pulp Fiction
Sylvester 2009
friend
Homecoming 09
organized
attended
type
attended
Fundraiser for School
type
organized
Gone with the wind
likes
Bob
Thriller
Jessie
attended
Alice Goodbye
Chill-out Night
likes
friend
likes
attended
type
Mrs. Doubtfire
organized
attended
likes
organized
Inception
Melissa
friend
Spring Break Trip
type
Alice
attended
likes
likes
friend
organized
Comedy
likes
likes
Emily
Harry Potter
likes
The Lion King
type
likes
type
Jon
type
likes
Mystery
Toy Story
type
Family
likes

Example Query



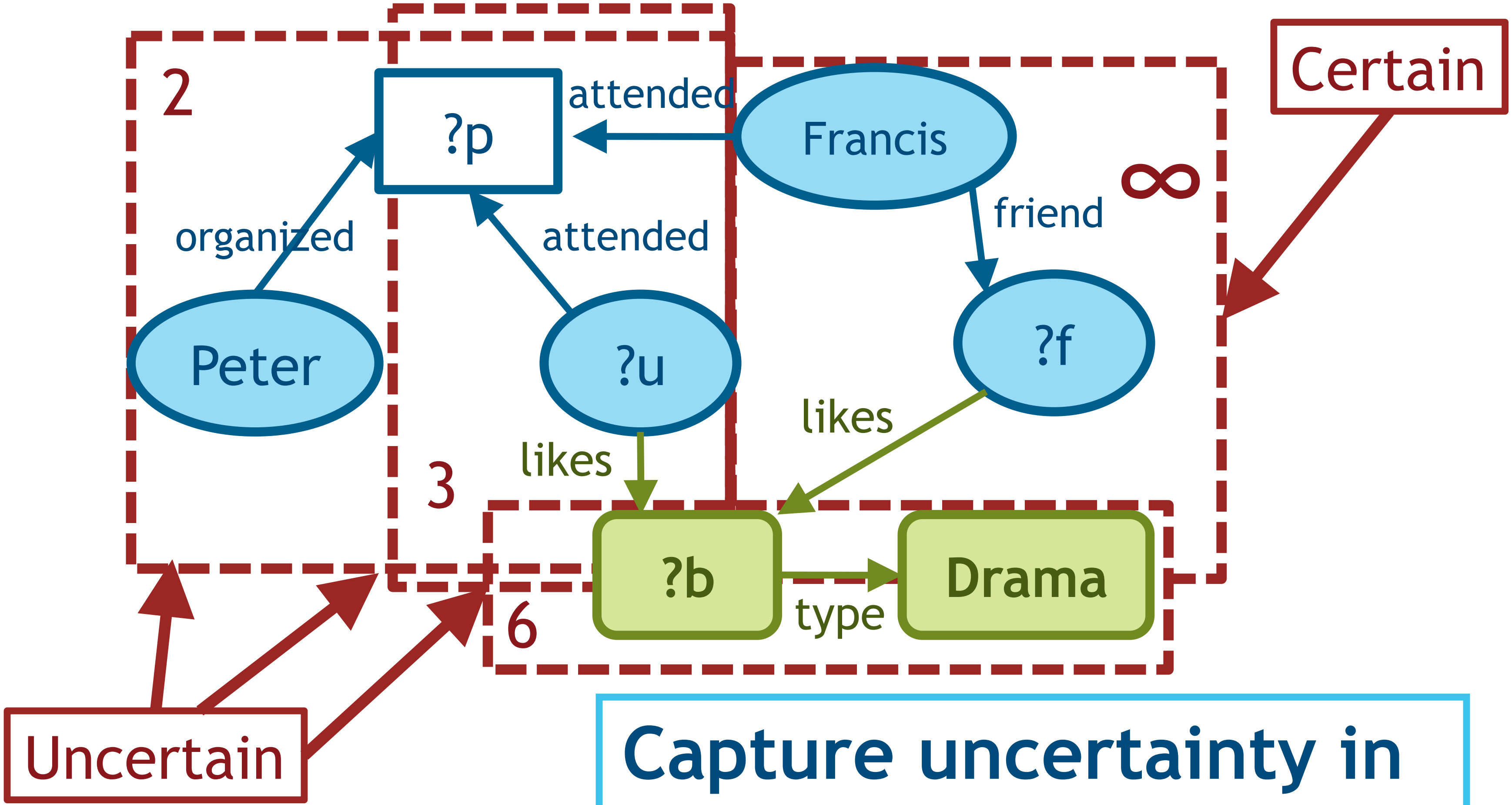
Simple query, yet complex structure and difficult to answer by hand

Query Uncertainty



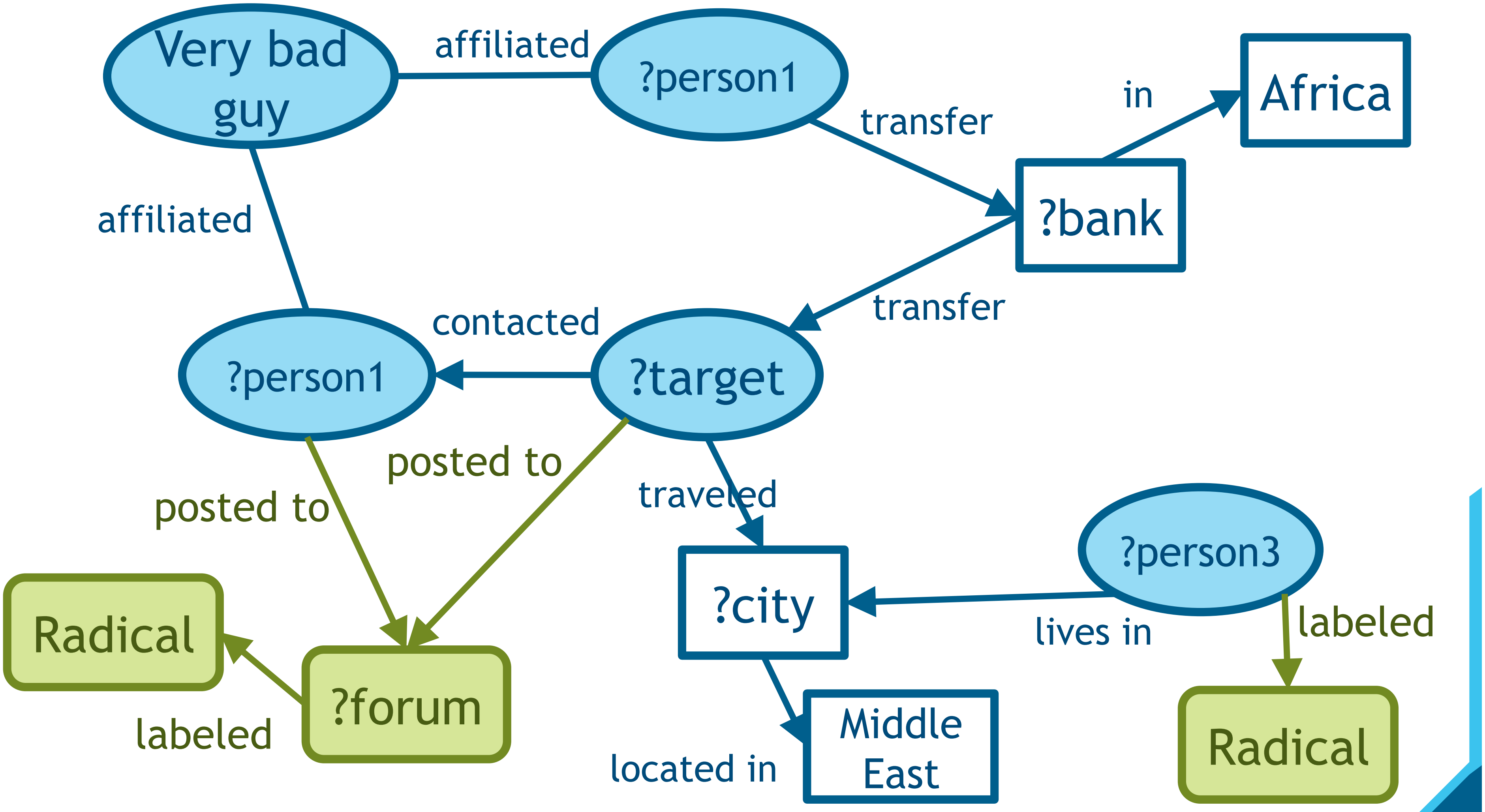
Was it really a drama movie?

Probabilistic Matching Query

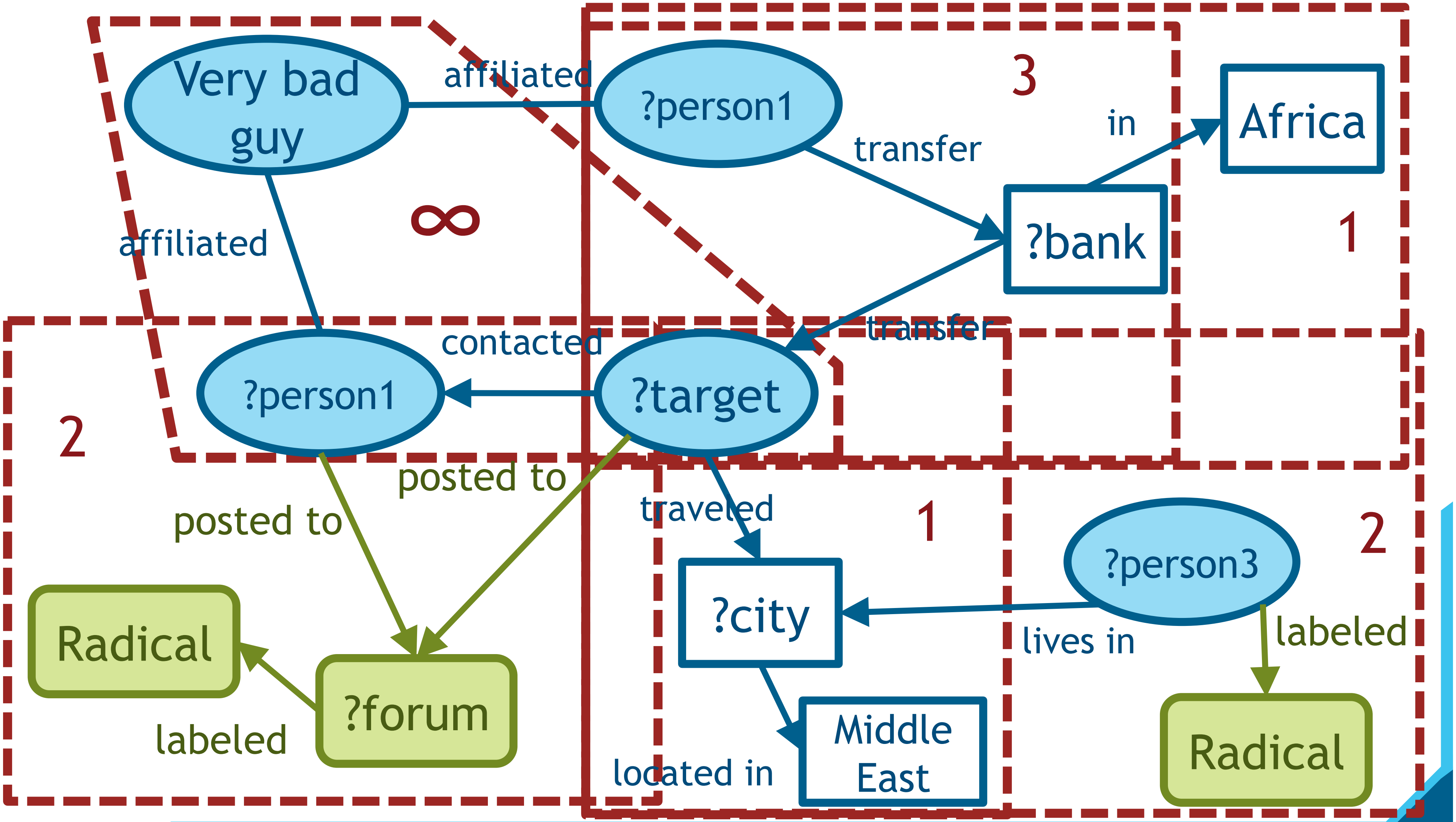


Capture uncertainty in weighted query boxes

Intelligence and Security



Intelligence and Security



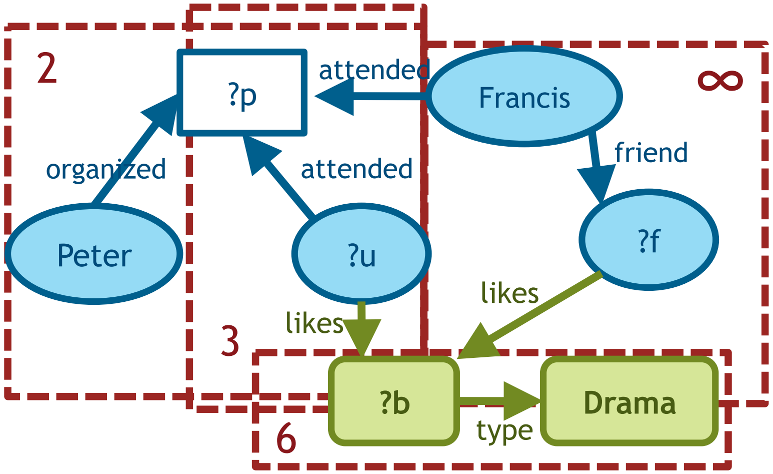
PMatch Applications

- Intelligence
  - Information retrieval on open source data
- Search
  - Facilitate productive use of social network data
- Security
  - Finding suspicious patterns
- Social Network Analysis
  - Querying as a primitive operation for data retrieval

Outline

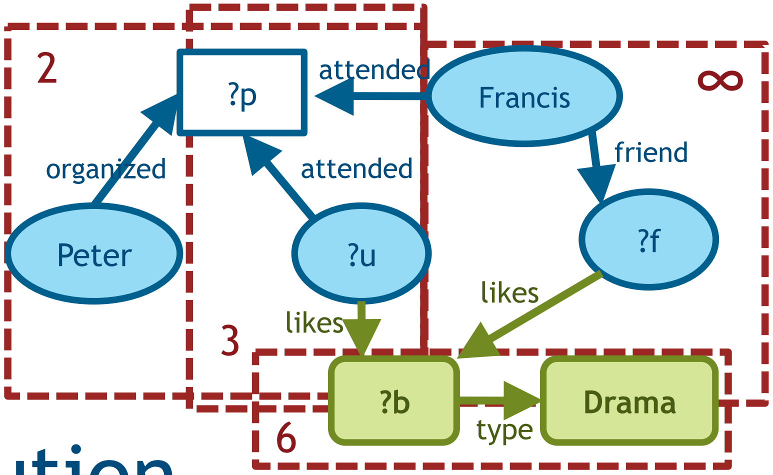
- Motivation
- PMatch Query Definition
- PMatch Query Answering
- Experiments
- Related Work & Conclusion

PMatch Query



- Set of Boxes
  - Each contains edges
  - Has a weight
    - Infinity → certainty or "core" of query
- Naturally extends exact subgraph matching queries

PMatch Query



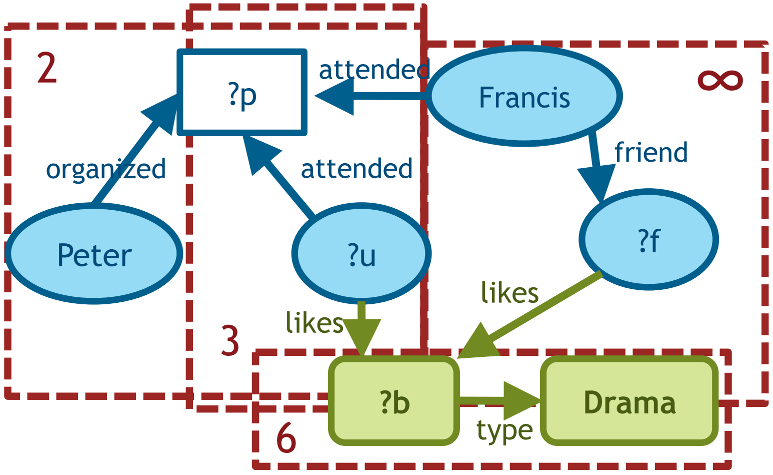
- Matching boxes
  - Let b_i = 1 if box B_i is matched by a substitution else 0 for i=1 to n (= # of boxes)
  - Weights w_i, offset w_0
- Probability of match (user defined)

  =0 if b_0 = 0, core must be matched

  = $$\frac{1}{1+e^{-(w_0+\sum_{i=1}^{n} w_i b_i)}}$$

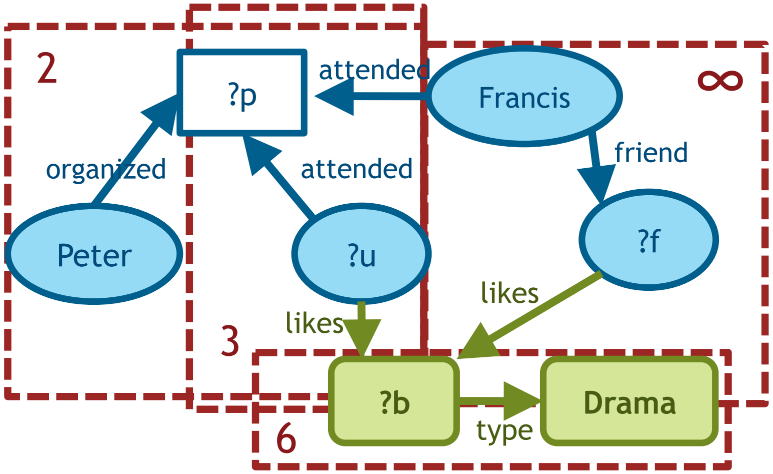
PMatch Query Answer

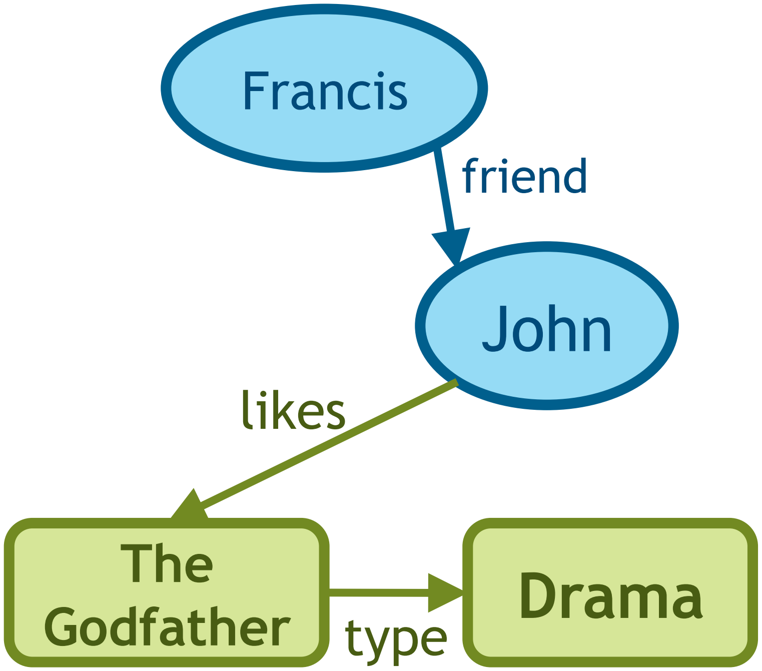
- All substitutions with probability above user defined threshold τ
- Need to eliminate redundant answers
  - Substitutions that can be extended to higher probability



PMatch Answer

- $1/(1+e^{-(-5+6)}) \approx 73\%$
  - w_0=-5





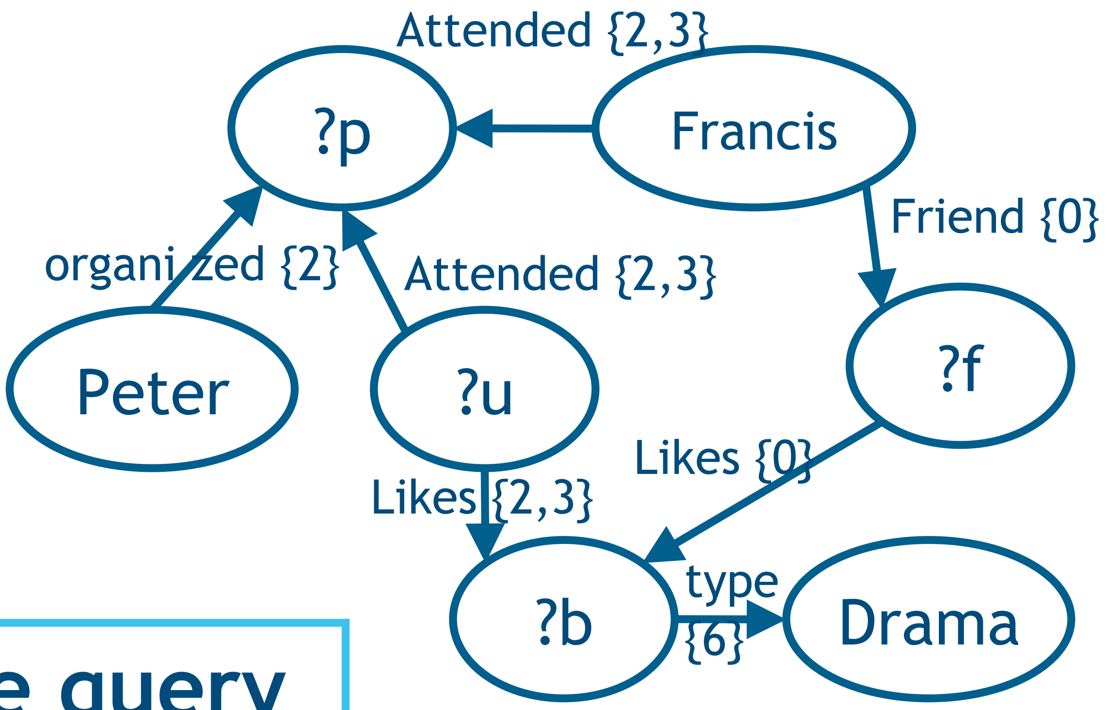
Outline

- Motivation
- PMatch Query Definition
- PMatch Query Answering
- Experiments
- Related Work & Conclusion

PMatch Query Answering

- Try to find all answers at once by exploiting shared substructures and efficiently maintain data structures.
- Depth first-search algorithm
  - Terminates when search space is exhausted or answer probability falls below threshold.
- Provably correct

PMatch Initialization



Rewrite query

(0=certain)

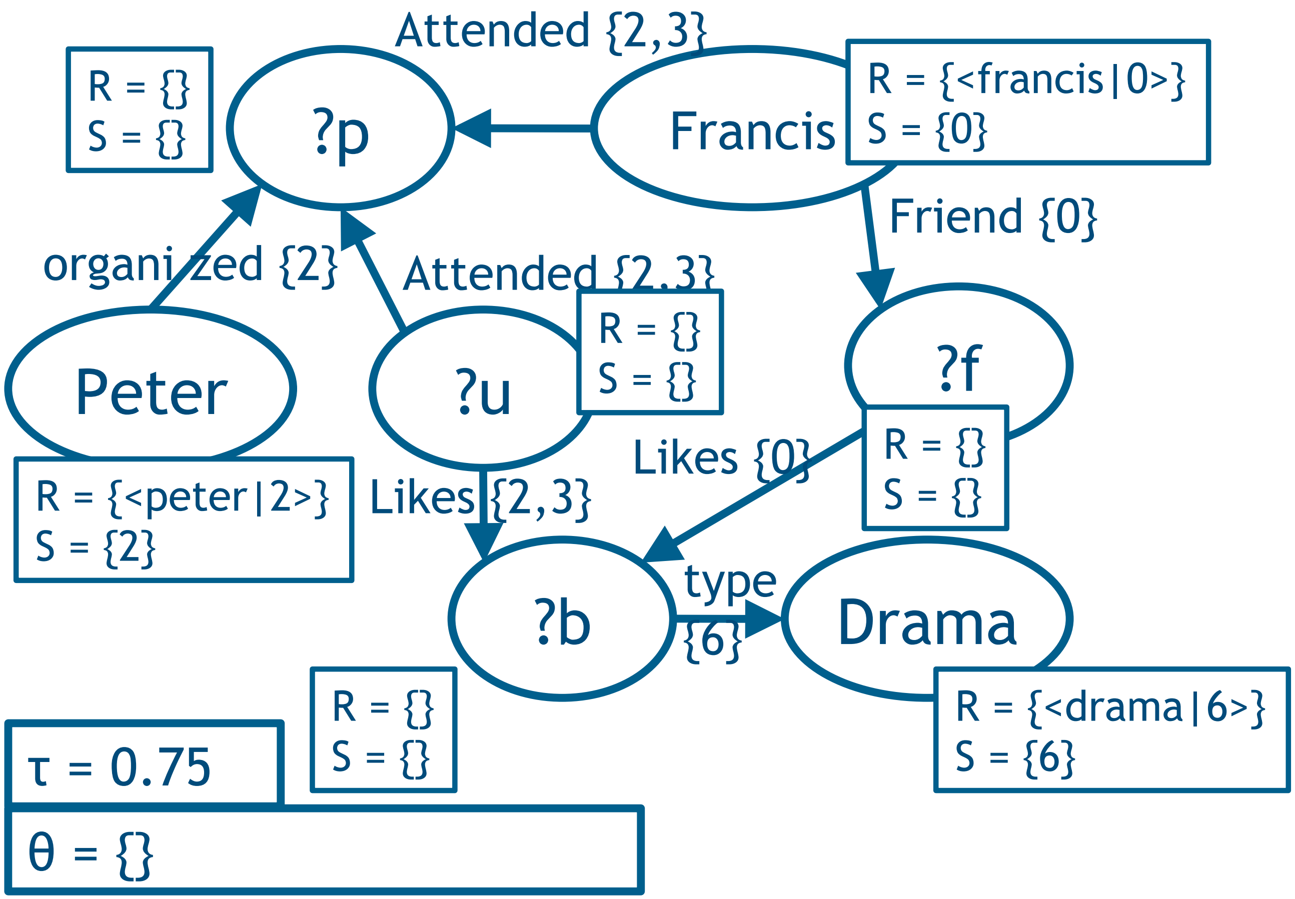
PMatch Data Structures

Maintained per vertex:

- R: Set of substitution candidates
  - Pairs of the form <x,H_x>, where x is a potential substitution and H_x the set of box ids it matches.

    H_x maintained as bit vector
- S: Set of box ids for which R has been initialized

PMatch Initialization

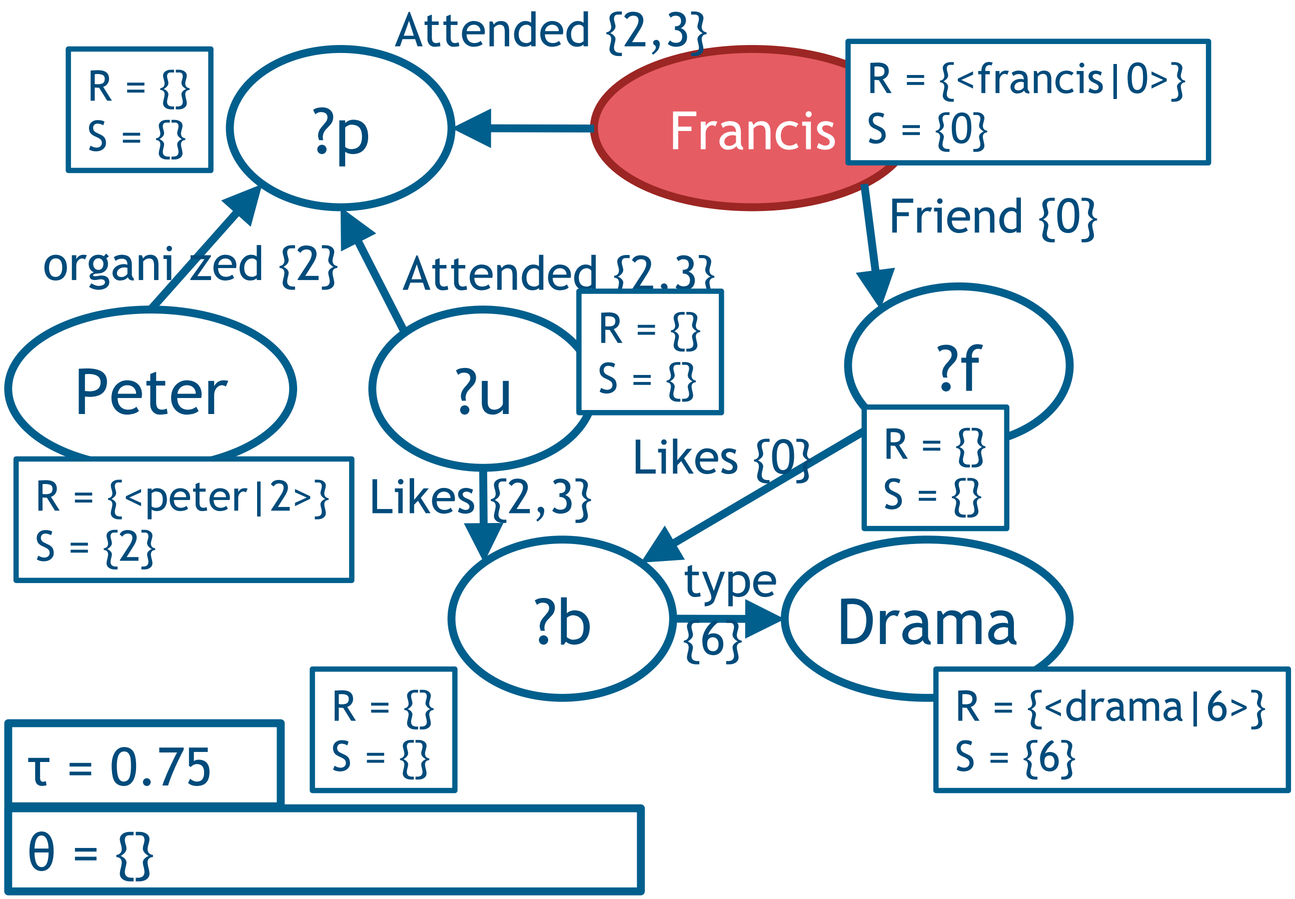


Vertex selection heuristic

- Goodness(v) = $\dfrac{\text{progress(v)}}{\text{cost(v)}}$
- Progress(v) = cumulative box weight of edges to be processed
  - Normalize weight by box cardinality
- Cost(v) = use standard cost models from exact subgraph matching
  - Cardinality, estimated selectivity

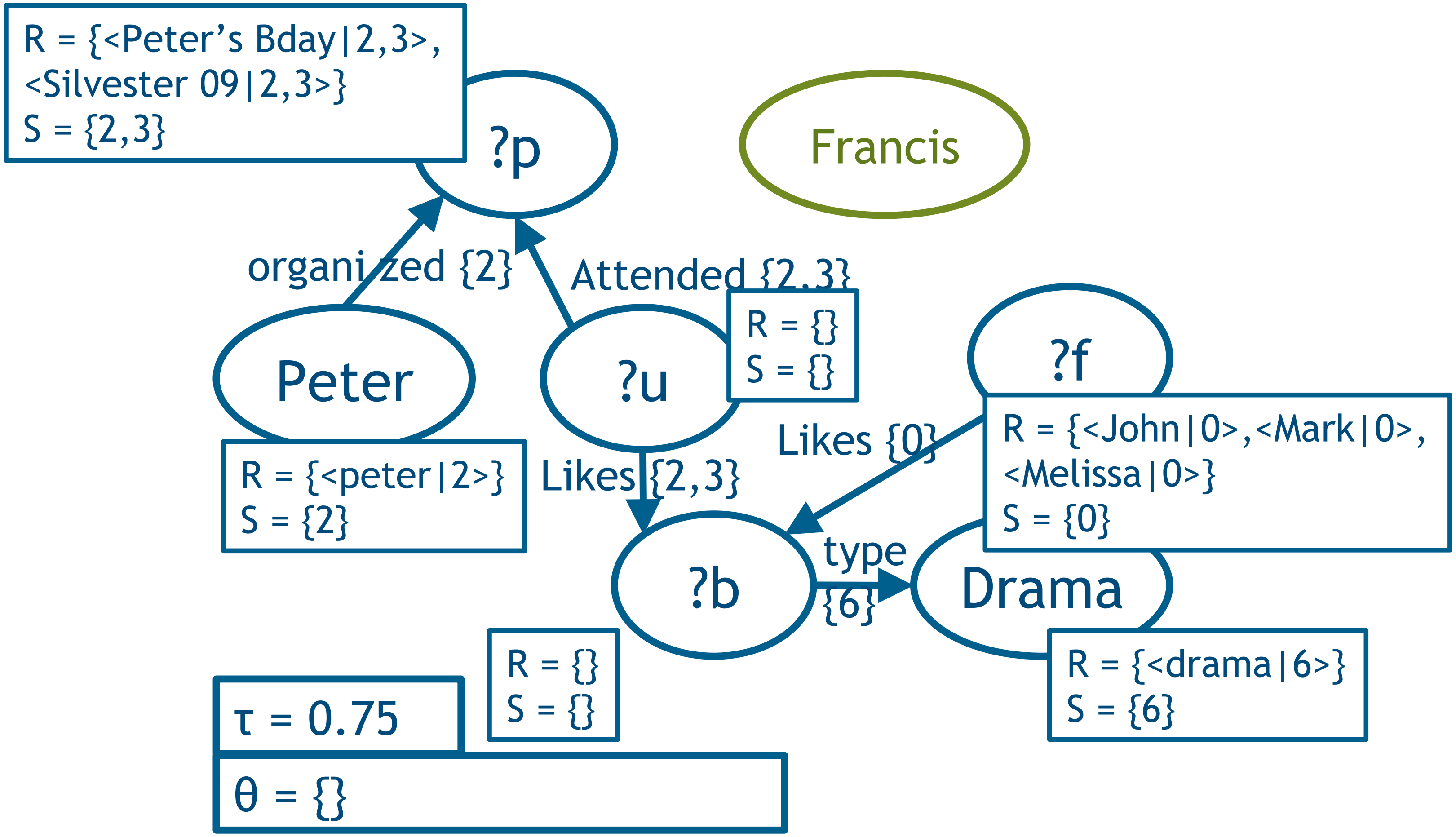
PMatch Algorithm

Initial vertex selection



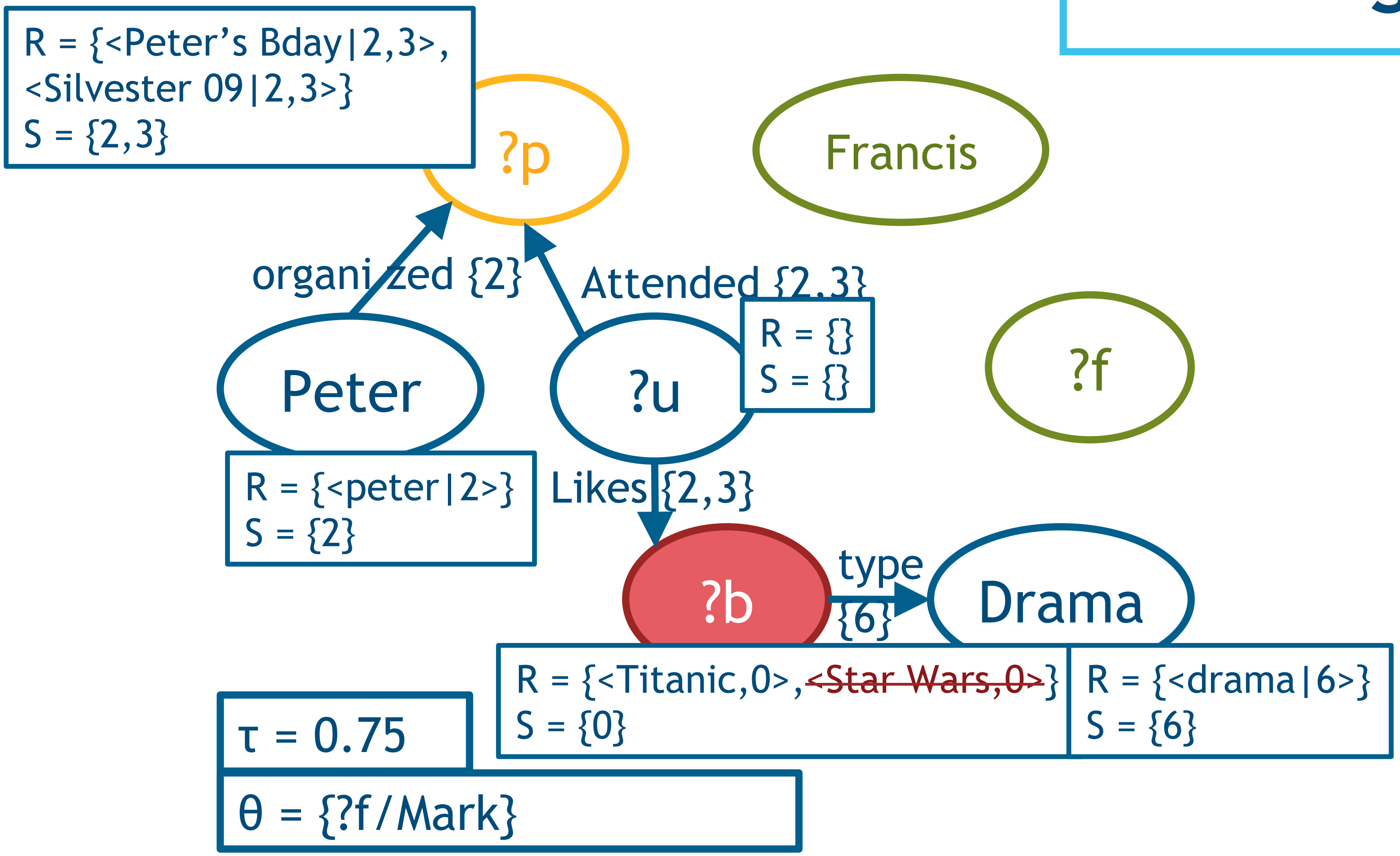
PMatch Algorithm

Processing of „Francis“ vertex



PMatch Algorithm

Threshold Pruning



R = {<Peter’s Bday|2,3>, <Silvester 09|2,3>}
S = {2,3}

R = {}
S = {}

R = {<peter|2>}
S = {2}

R = {<Titanic,0>,~~<Star Wars,0>~~}
S = {0}

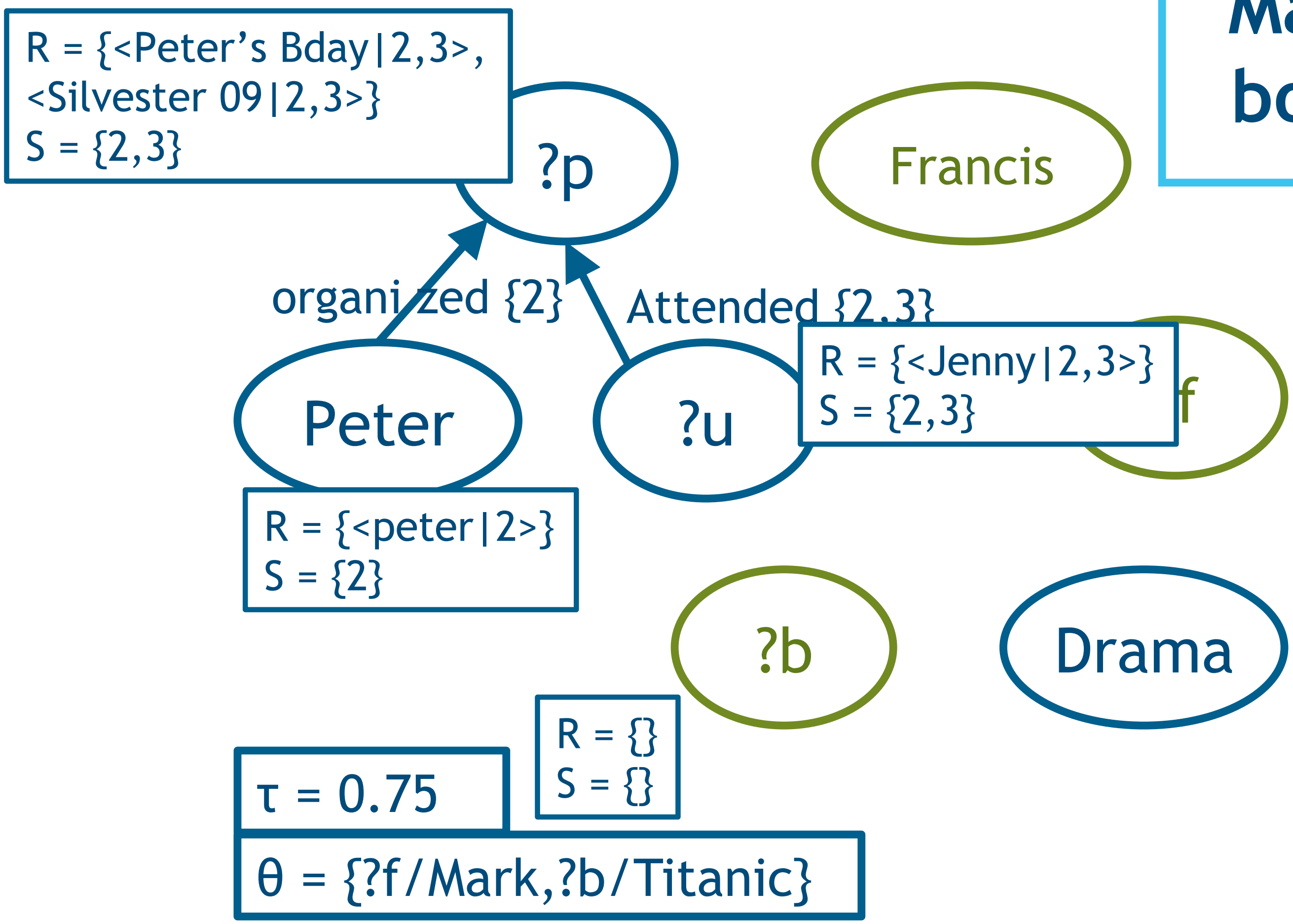
R = {<drama|6>}
S = {6}

τ = 0.75

θ = {?f/Mark}

PMatch Algorithm

Processing vertex „?b“. Maintaining box indices



R = {<Peter’s Bday|2,3>, <Silvester 09|2,3>}
S = {2,3}

?p

Francis

organized {2}

Attended {2,3}

Peter

?u

R = {<Jenny|2,3>}
S = {2,3}

f

R = {<peter|2>}
S = {2}

?b

Drama

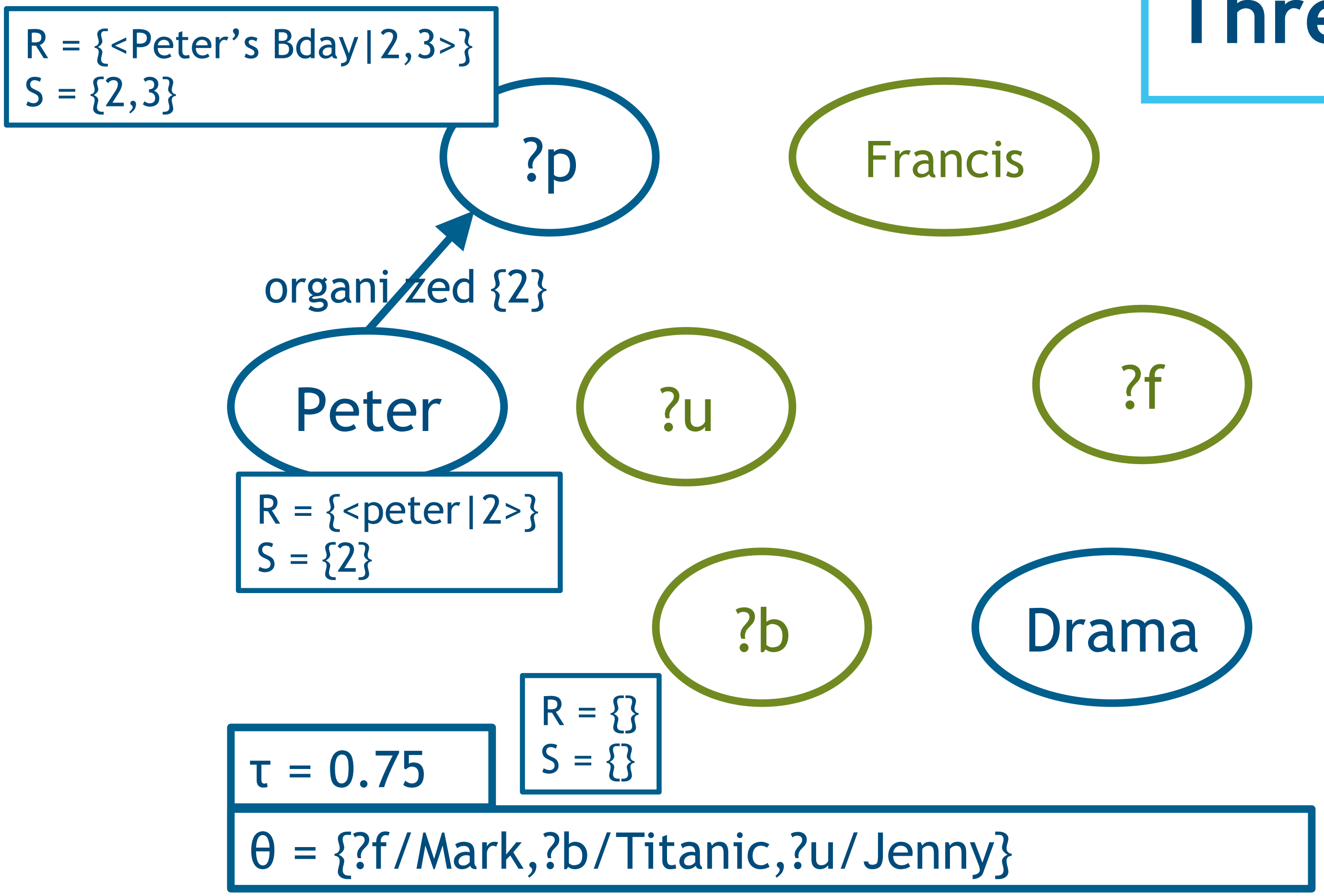
R = {}
S = {}

τ = 0.75

θ = {?f/Mark,?b/Titanic}

PMatch Algorithm

Crossed Threshold



R = {<Peter’s Bday|2,3>}
S = {2,3}

?p

Francis

organi zed {2}

Peter

?u

?f

R = {<peter|2>}
S = {2}

?b

Drama

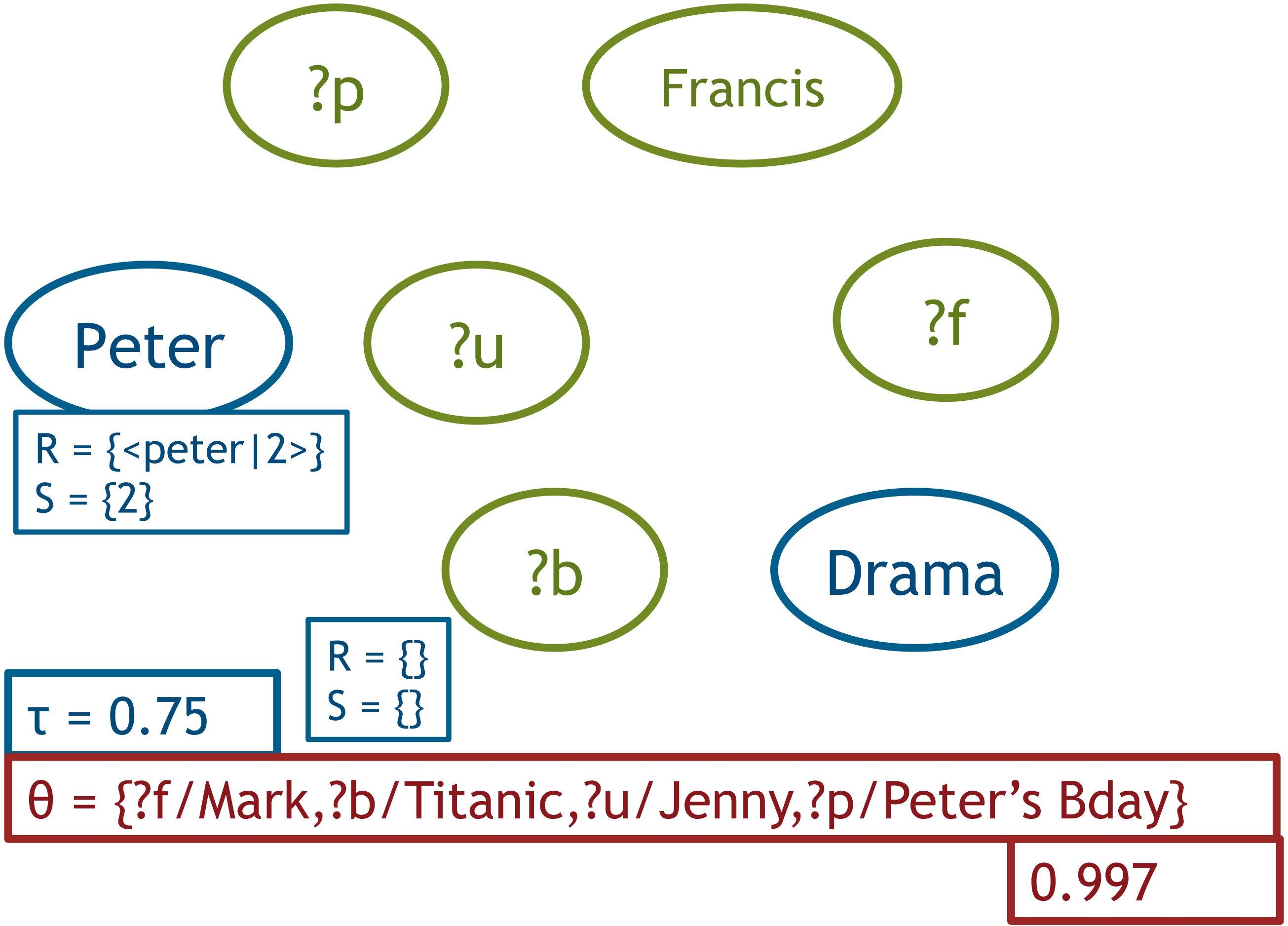
R = {}
S = {}

τ = 0.75

θ = {?f/Mark,?b/Titanic,?u/Jenny}

PMatch Algorithm

Found Result



R = {<peter|2>}
S = {2}

R = {}
S = {}

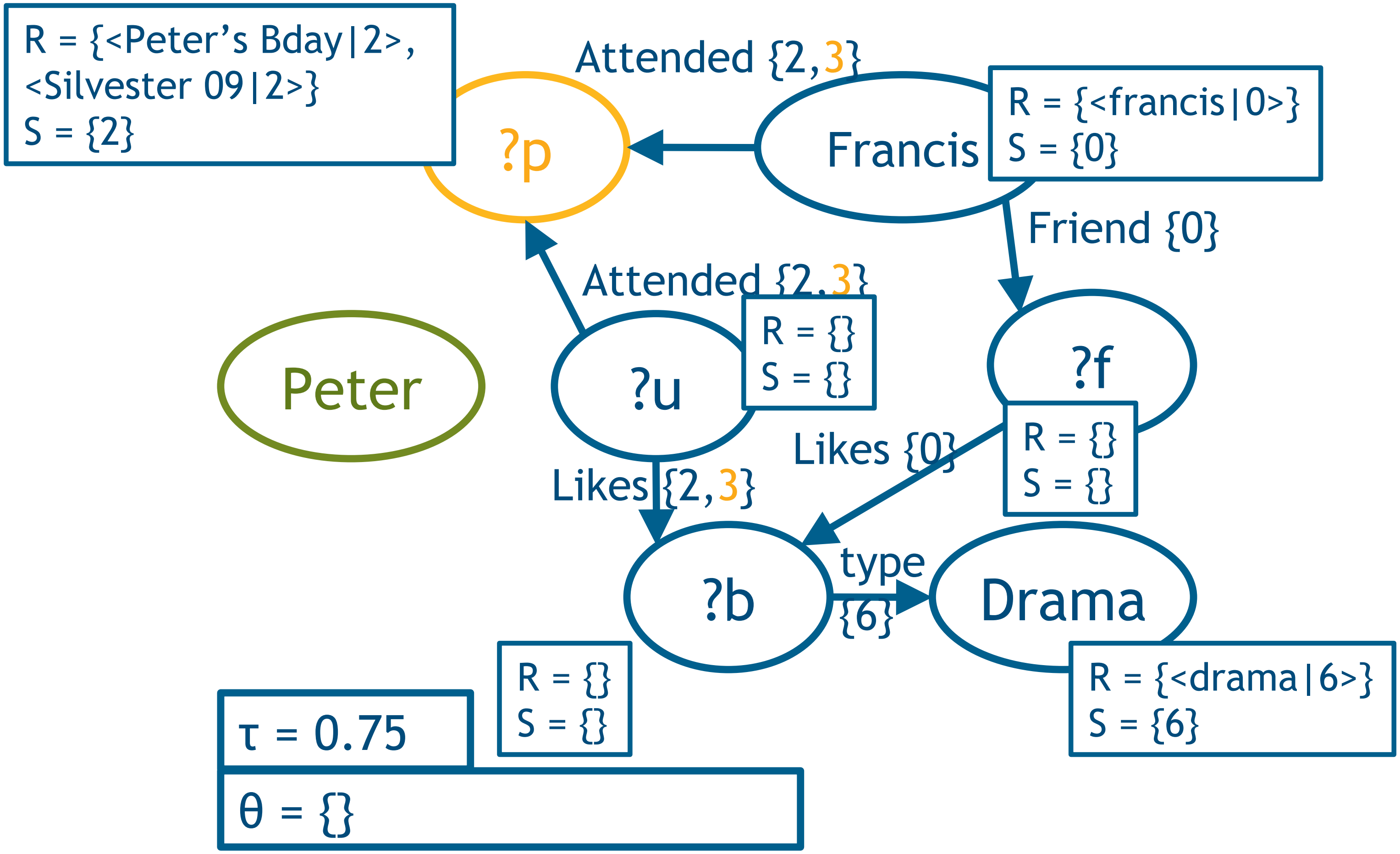
τ = 0.75

θ = {?f/Mark,?b/Titanic,?u/Jenny,?p/Peter's Bday}

0.997

Query Splitting

Selecting „Peter“ first causes query to be split.



R = {<Peter’s Bday|2>, <Silvester 09|2>}
S = {2}

?p

Attended {2,3}

Francis

R = {<francis|0>}
S = {0}

Friend {0}

Attended {2,3}

Peter

?u

R = {}
S = {}

?f

R = {}
S = {}

Likes {0}

Likes {2,3}

?b

type {6}

Drama

R = {}
S = {}

R = {<drama|6>}
S = {6}

τ = 0.75

θ = {}

Outline

- Motivation
- PMatch Query Definition
- PMatch Query Answering
- Experiments
- Related Work & Conclusion

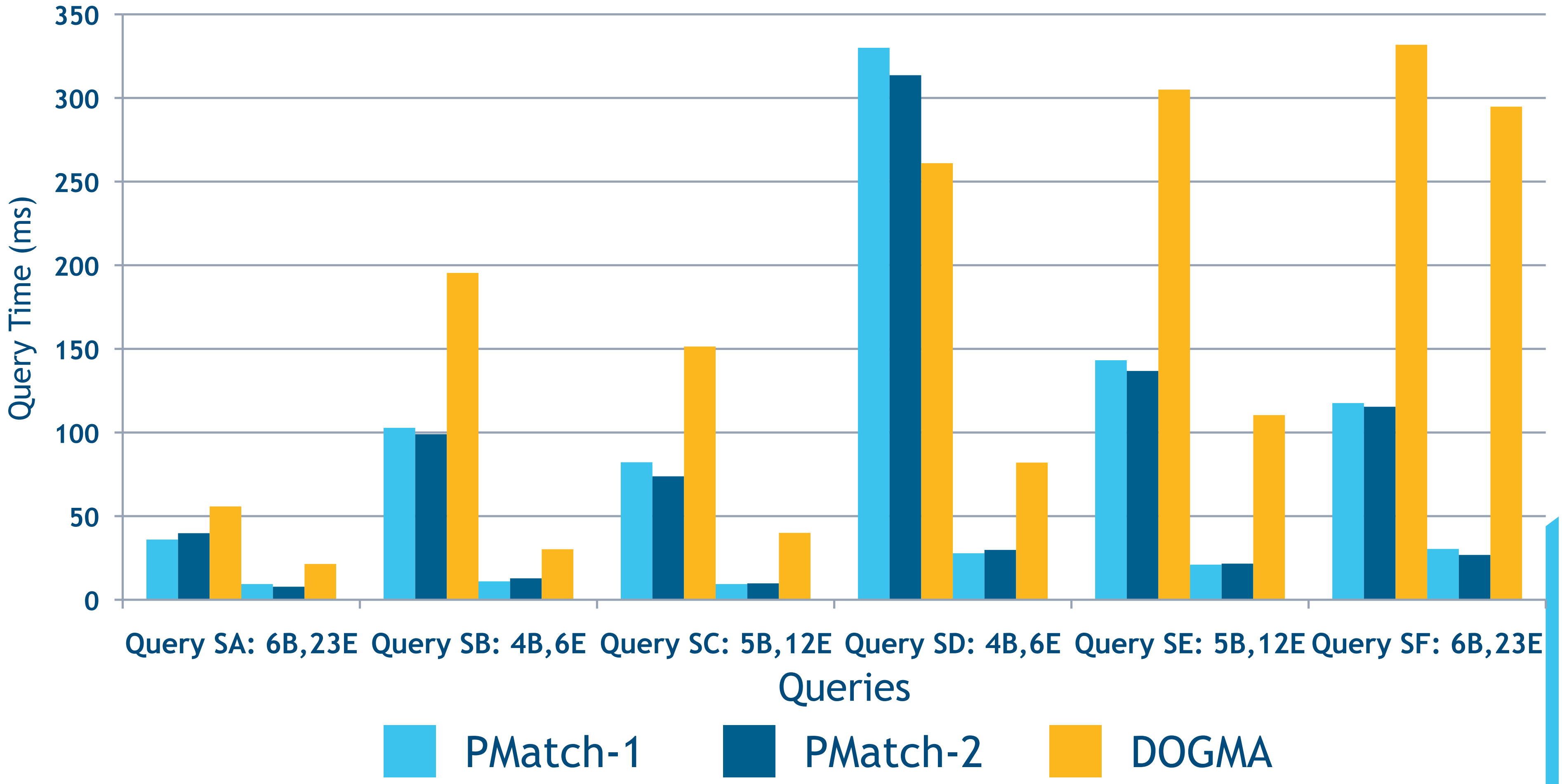
Experimental Setup I

- Implementation in Java (approx 5500 loc) on top of Neo4j
- Datasets
  - Friendship Network (Flickr, Orkut, Livejournal, Youtube): 778 million edges
  - Delicious Network: 1.1 billion edges
- Query Benchmark for each dataset: 9 PMatch queries each of increasing complexity

Experimental Setup II

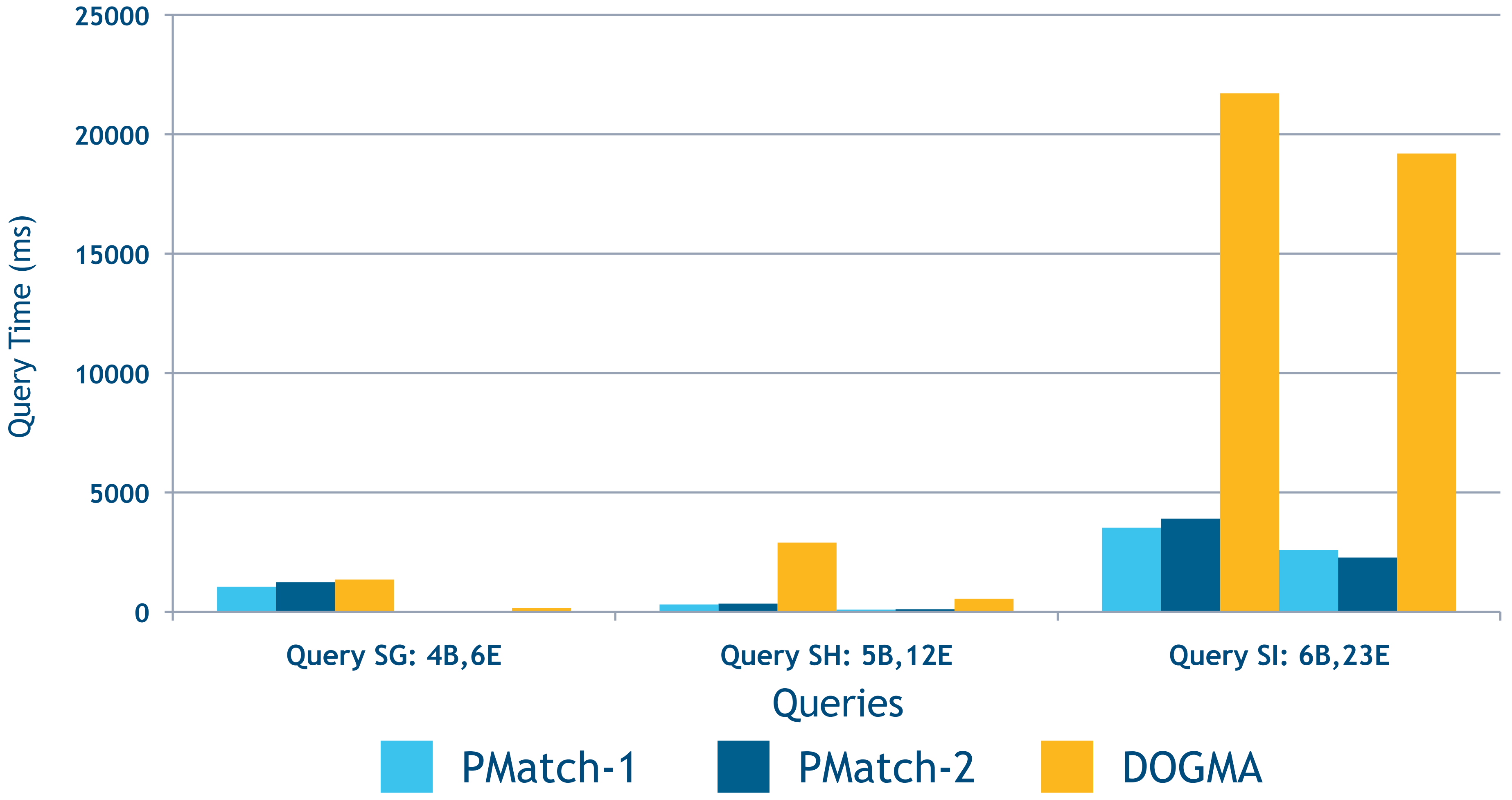
- Compared
  - PMatch-1: box weight
  - PMatch-2: normalized box weight
  - DOGMA: Exact matching for all combination
  - Neo4j: Exact matching for all combinations
    - Too slow – not shown in the charts
- System: AMD Opteron, 15K RPM 300 GB SAS HD, 2GB of heap space

Friendship network (small queries)



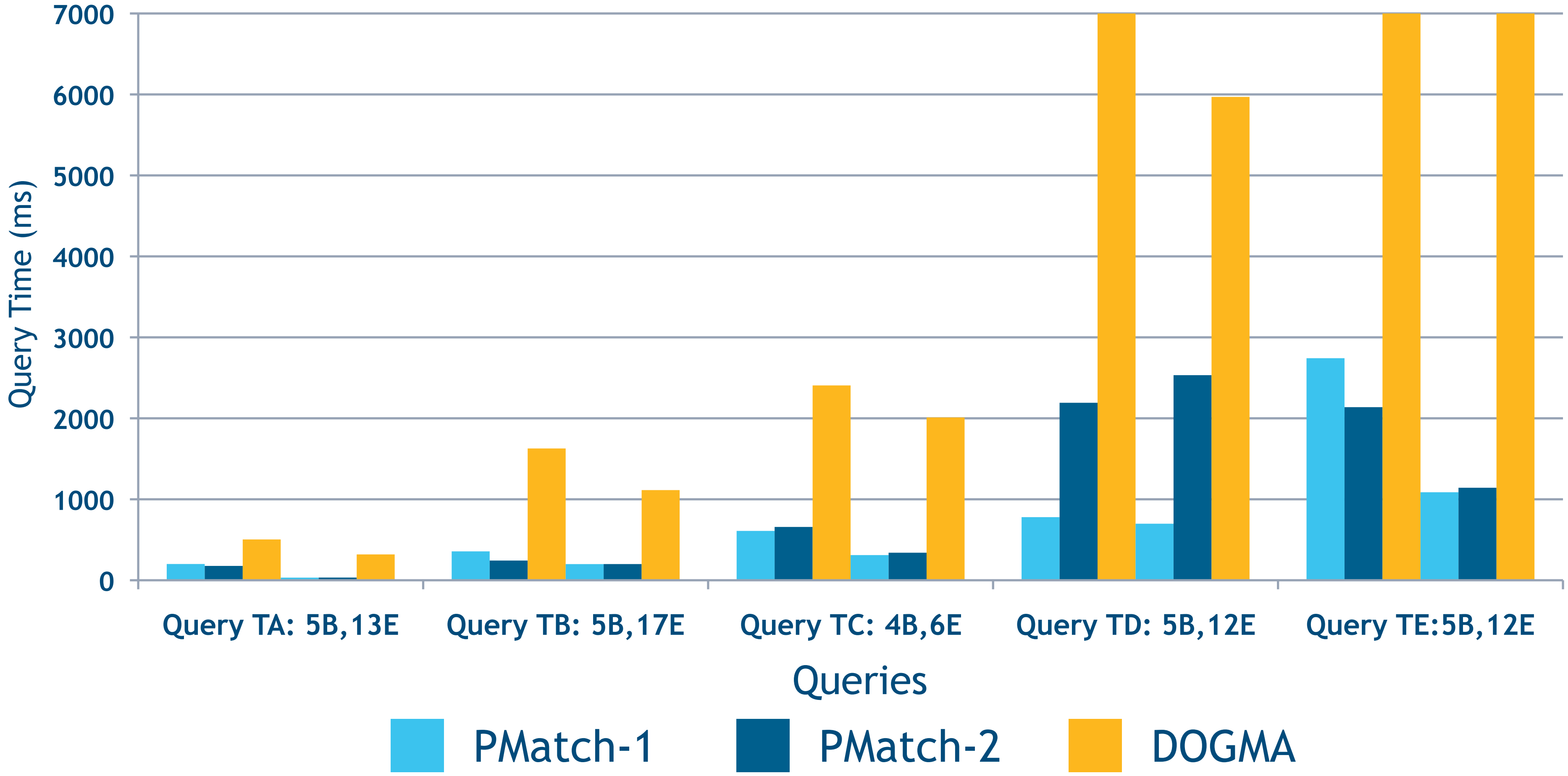
For each query, we report query times with cold (right) and warm (left) caches.

Friendship network (large queries)



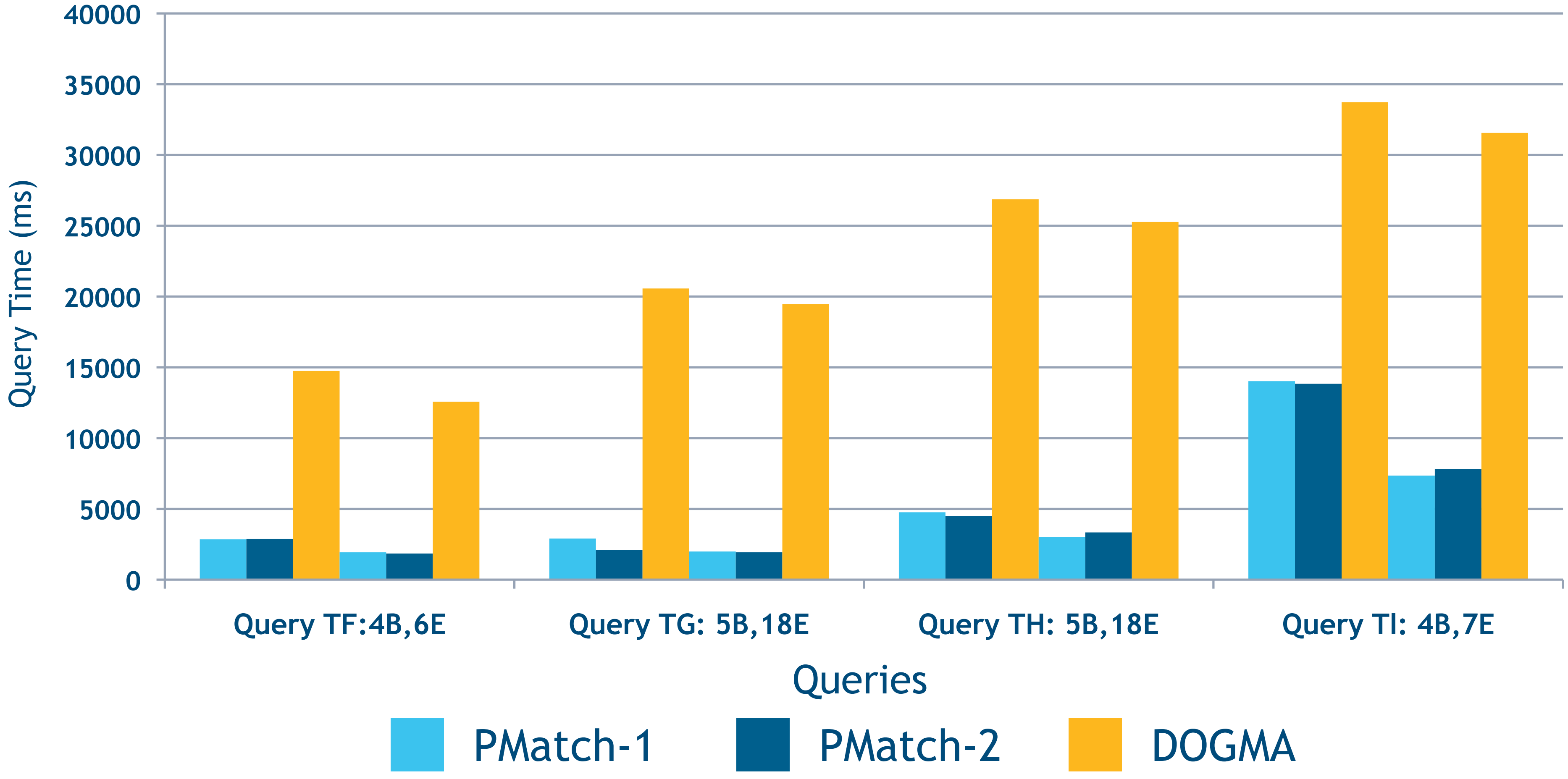
For each query, we report query times with cold (right) and warm (left) caches.

Delicious network (small queries)



For each query, we report query times with cold (right) and warm (left) caches.

Delicious network (large queries)



For each query, we report query times with cold (right) and warm (left) caches.

Outline

- Motivation
- PMatch Query Definition
- PMatch Query Answering
- Experiments
- Related Work & Conclusion

Related Work I

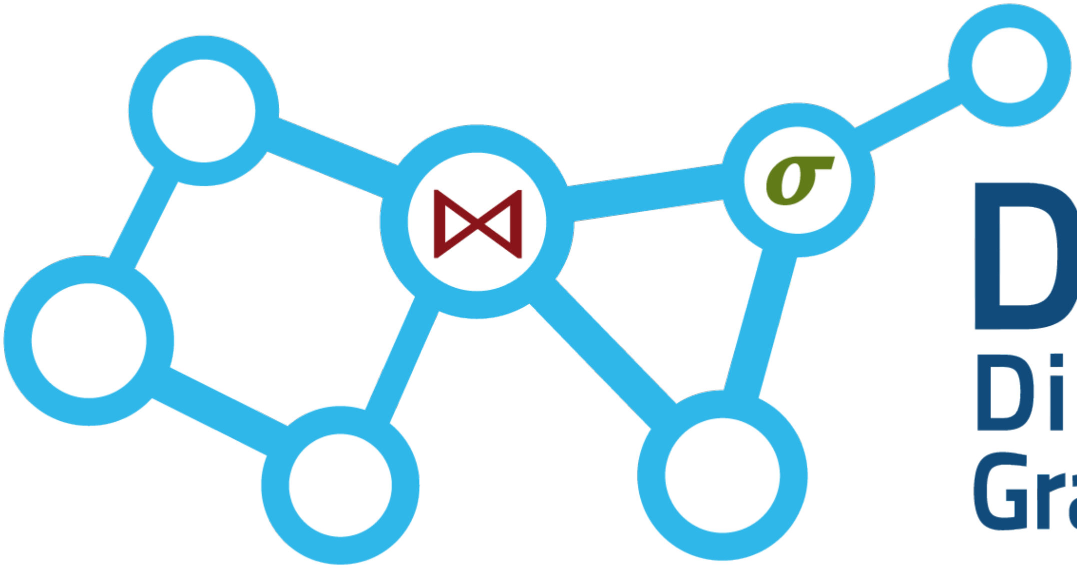
- Exact Subgraph Matching
  - Many systems: RDF-3X, OWLIM, YARS (2), Sesame, Jena, DOGMA, COSI etc.
  - Efficient on-disk index structures
  - Query answering algorithms focused on exact queries (e.g. SPARQL).
- No treatment of query uncertainty

Related Work II

- Approximate Subgraph Matching
  - SAGA, TALE, Grafil, etc
  - Database containing many small graphs
  - Do not scale to large social networks
  - Query uncertainty is defined in the system.
- PMatch provides an expressive language to specify probabilistic queries and answers them efficiently.

Conclusion

- Query uncertainty often occurs in social network search, information retrieval and pattern matching.
- PMatch provides a language to concisely express and algorithms to efficiently answer probabilistic subgraph matching queries.



DOGMA

Disk Oriented Graph Matching

dogma.umiacs.umd.edu

Questions?
Comments?

Related Work III

- Data Uncertainty
  - Specified at the tuple or edge level
  - How to answer certain (SQL) queries over uncertain data and what are the answer probabilities?

First Name	Last Name	Income	Probability
John	Smith	80,000	0.6
Alice	Baker	85,000	0.5
Jennifer	Temper	90,000	0.9

Related Work III

- Data Uncertainty
  - How to encode dependencies?
  - Need to make strong independence assumptions to be efficient
  - Where are the probabilities coming form?
- Instead, we assume certain data and model uncertainty in the user specified query.

Query Uncertainty Motivation

- User Uncertainty
  - User does not know what she is looking for exactly
- Lack of schema
  - Lack of schema complicates query design
- Data heterogeneity
  - Can be caused by data integration
- Noisy or Missing Data
  - It’s real world data