The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications

May 24 - 29, 2015 - Rom, Italy

# Overview of Regular Path Queries in Graphs


Andreas Schmidt[1,2], Iztok Savnik[3]

(1)
Department of Informatics and
Business Information Systems
University of Applied Sciences Karlsruhe
Germany

(2)
Institute for Applied Sciences
Karlsruhe Institute of Technologie
Germany

(3)
Department of Computer Science
University of Primorska
Slovenia

# Outlook

- Introduction
- Application Fields
- Types of Graphs
- Regular Path Queries (RPQ)
- Extensions of RPQ

# Introduction

- Query languages for graph databases ...
  - are navigational/recursive
  - traverse the labeled edges/nodes
  - query the topology of the data, but not the data itself
- Basic building blocks for this languages are often regular path queries (RPQ)
- RPQ are used in many graph query languages, so i.e. in
  - G
  - GraphLog
  - Cypher
  - XPath
  - SPARQL 1.1

# Application Fields for Graphs and Regular Graph Queries

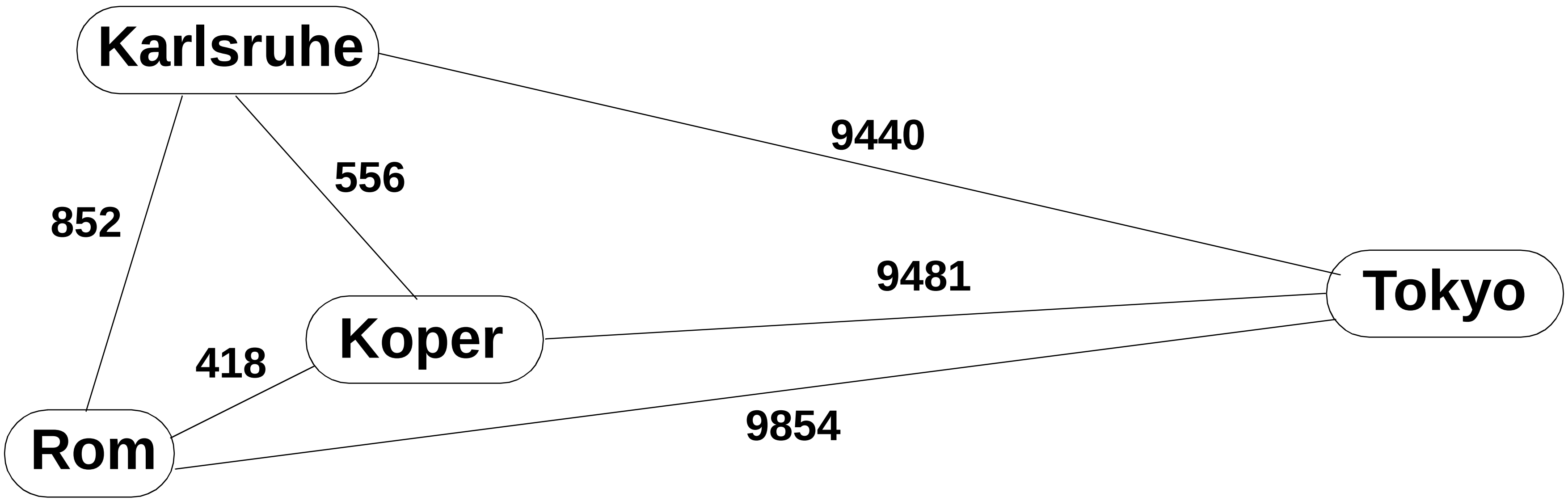
- Knowledge representation (RDF/s, OWL, Ontologies like Yago, Taxonomies)
  - connection between entities, instance and subclass relationships
- Transportation networks (airline, train, bus, streets)
  - Reachability, shortest path, critical path
- geographical information
- biological applications
- bibliographic citation analysis
- social networks
- program analysis
  - reachability of code, variables used before defined, deadlocks

# Query Types

- Graph Pattern Matching (subgraph matching)
- Path Finding (finding nodes connected by graphs)
- Extraction of edge label variables
- Aggregation
- ...

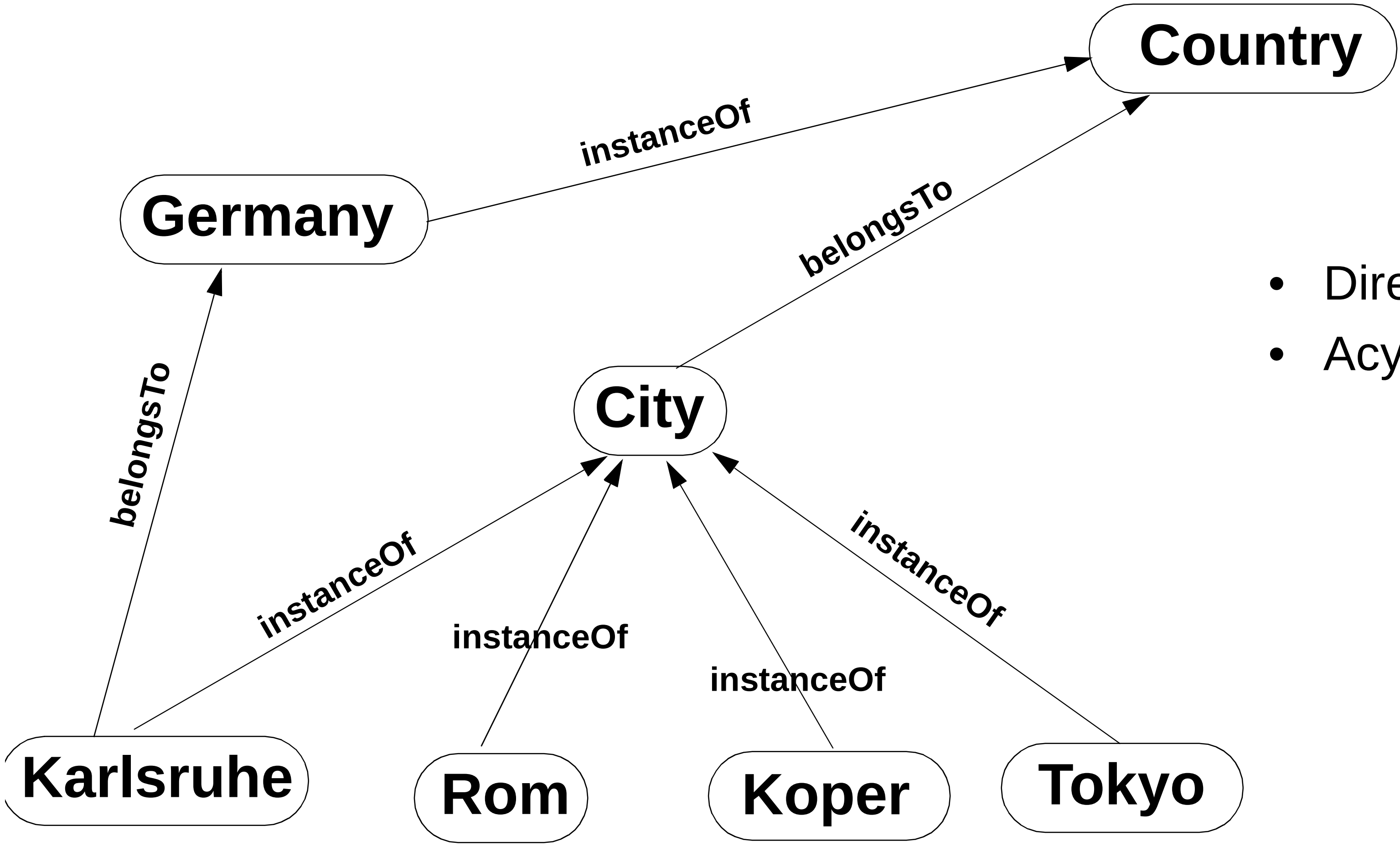
# Types of Graphs

- Undirected
- Cyclic




**Distances from http://www.luftlinie.org**

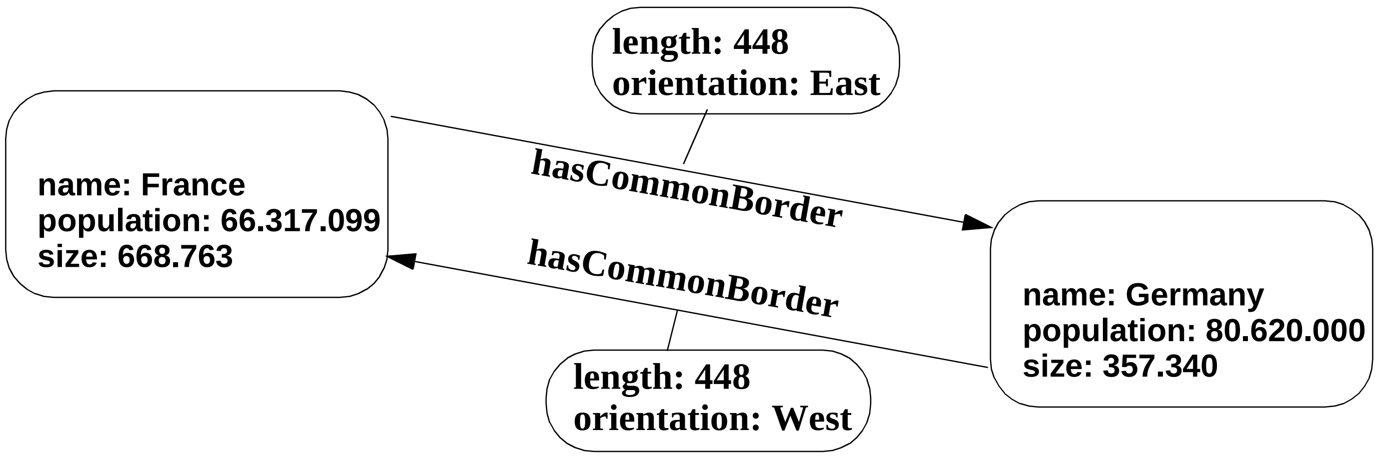
# Types of Graphs




- Directed
- Acyclic

# Property Graph

- directed
- cyclic
- Nodes and relations have properties



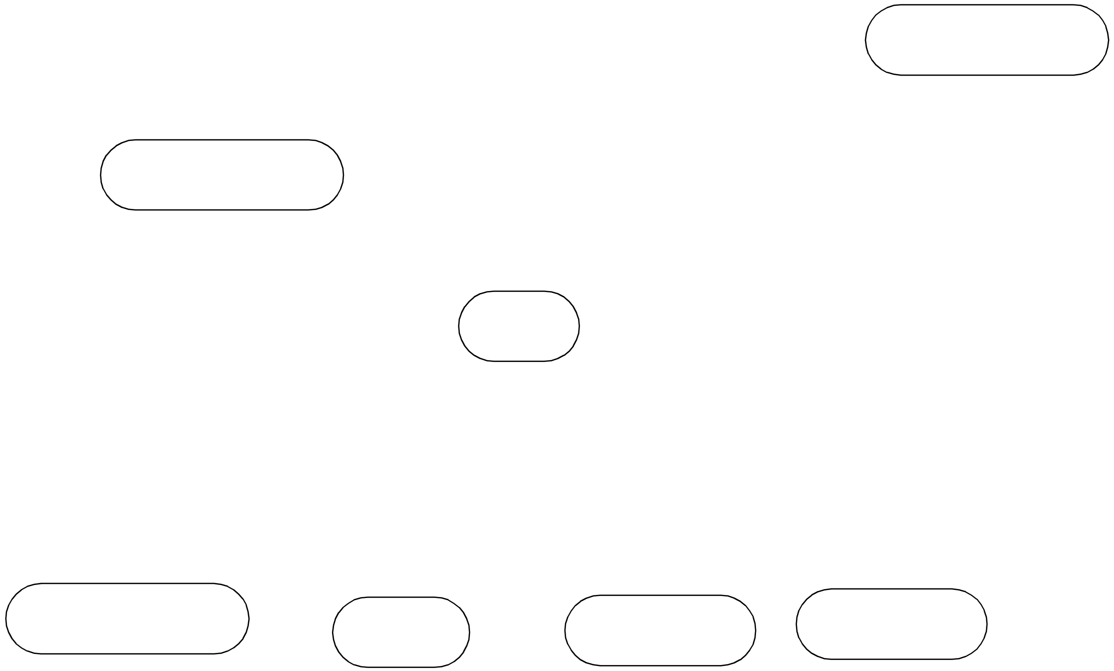
# Yet another Graph: Type ?

# Definition of Database Graph

G=(N, E)

Directed, labeled graph with:

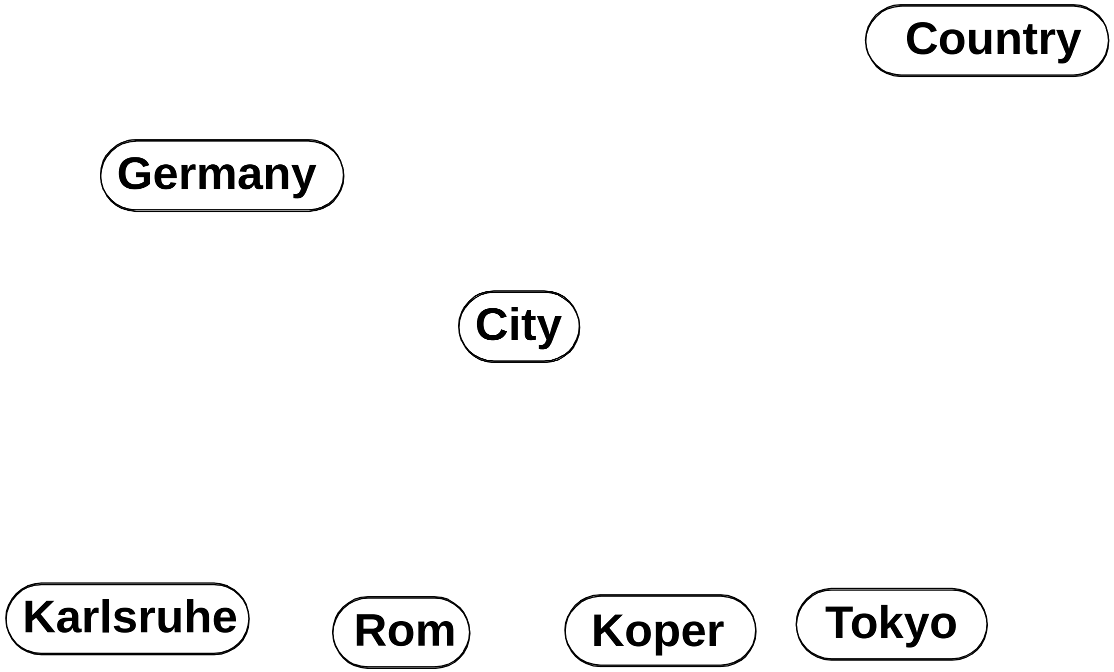
- N: set of nodes



G=(N, E)

Directed, labeled graph with:

- N: set of nodes, representing entities from the real world

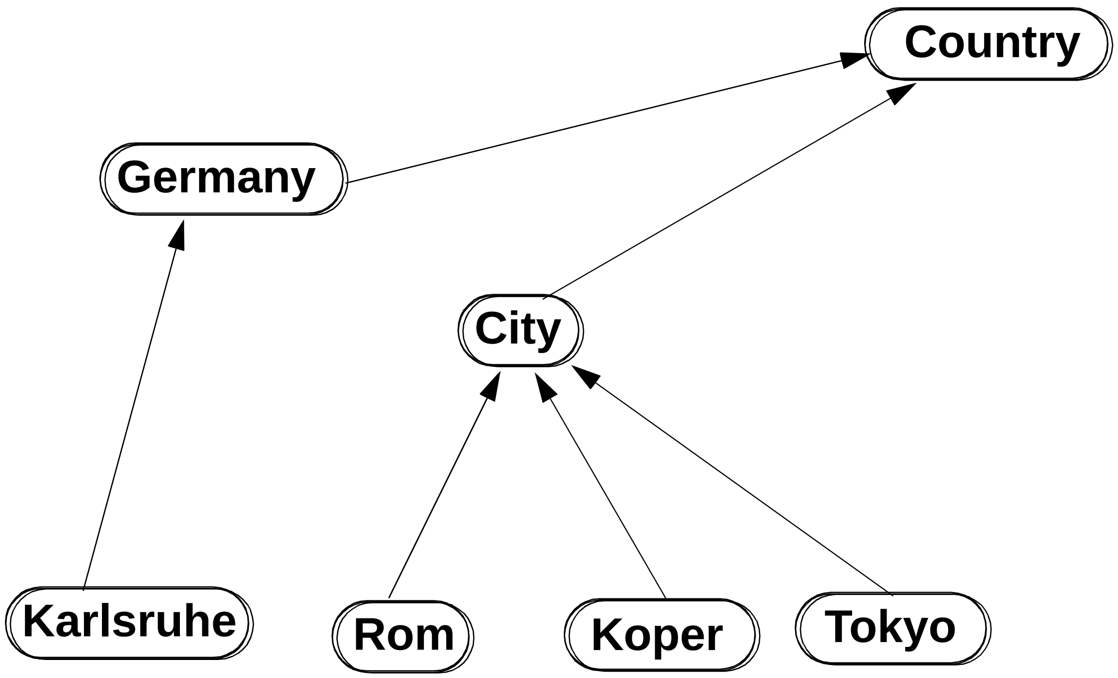


# Definition of Database Graph [MW89]

G=(N, E)

Directed, labeled graph with:

- N: set of nodes,, representing entities from the real world
- E: set of directed edges

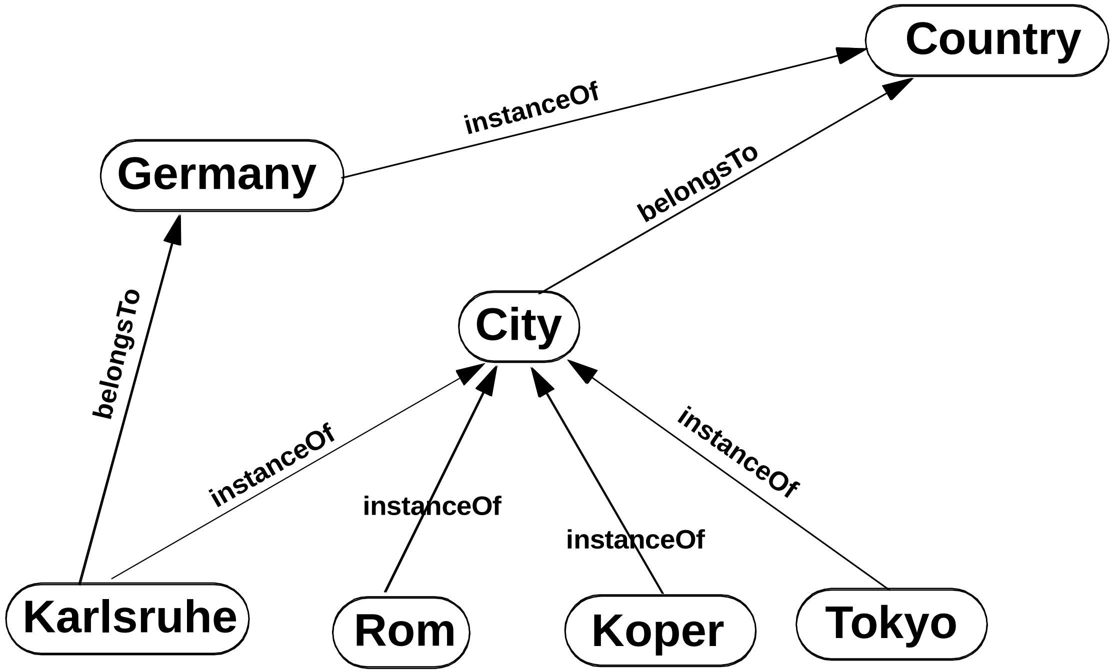


# Definition of a Database Graph for RQP

G=(N, E)

Directed, labeled graph with:

- N: set of nodes
- E: set of directed edges
- S: finite set of symbol for labeling of edges (vocabulary)



# Regular Path Queries (RPQ)

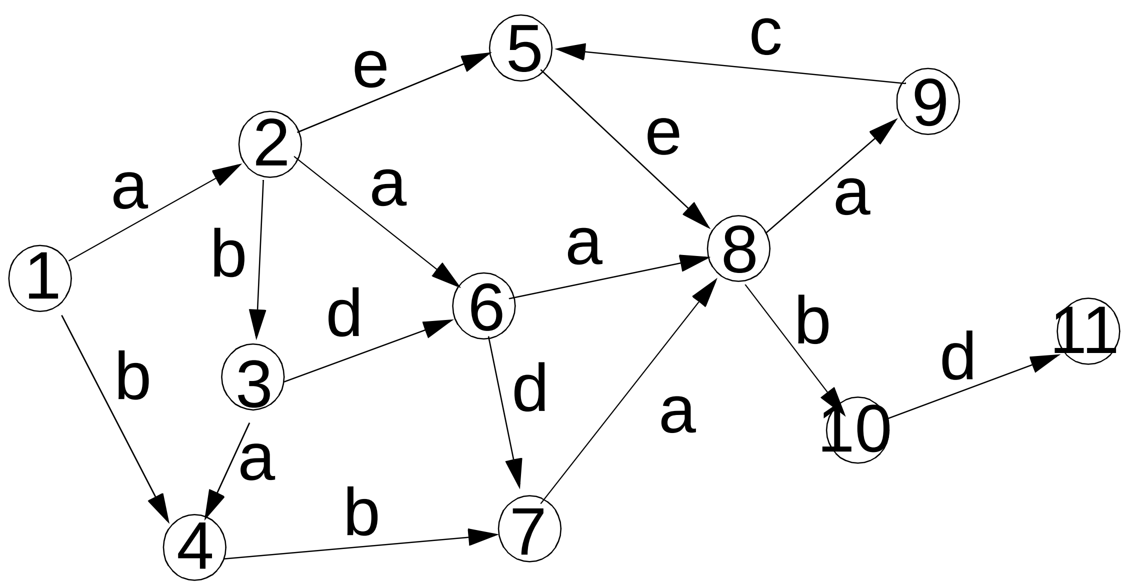
- RPQ have the form:
  RQP(x,y) := (x, R, y)
  where R is a regular expression over the vocabulary of edge labels

- Construction of regular expressions:
  R ::= s | R.R | R|R | R* | R? | (R)          // s element from S

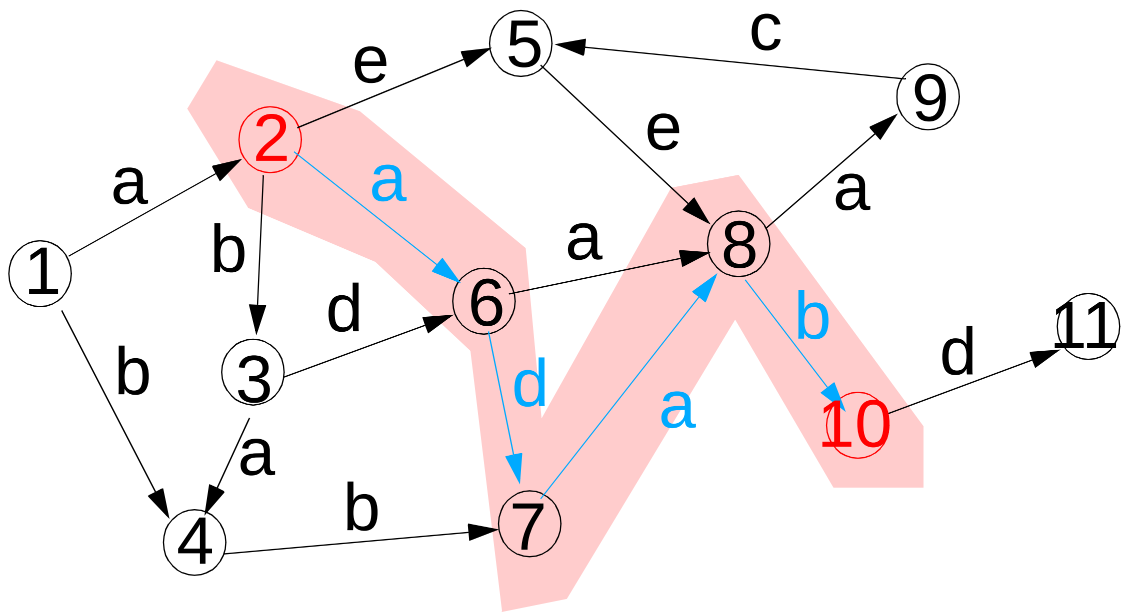
- Examples:
  - Ancestors:
    isChildOf+
  - Cousins
    isChildOf . isMarriedWith . isChildOf . hasChild . hasChild

# RPQ Example

$R = a+(d|b)ab$



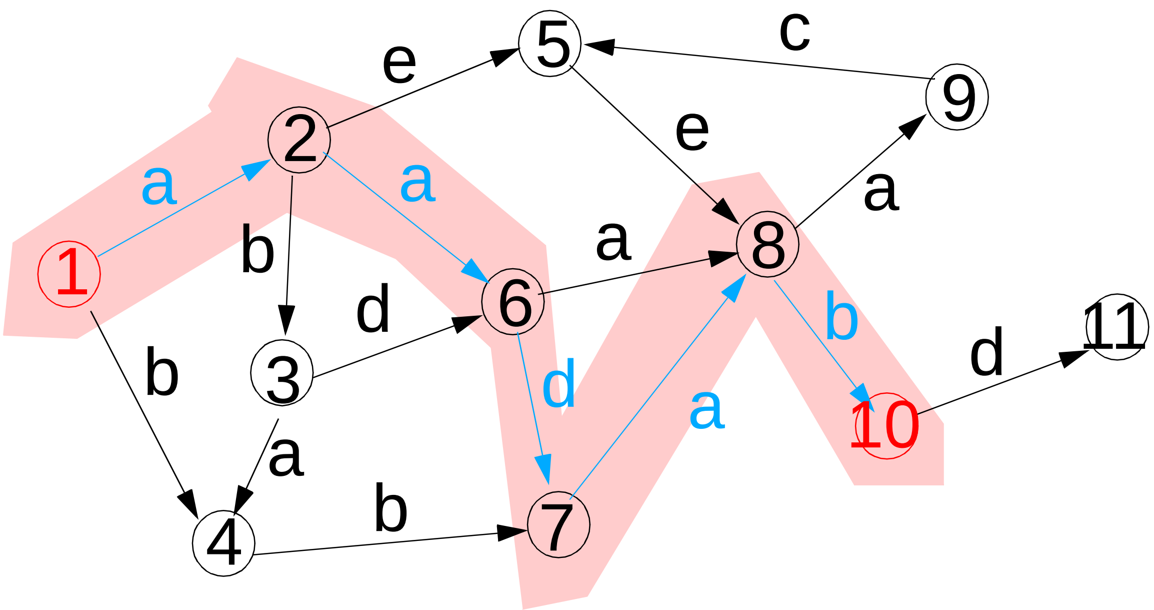
# RPQ Example




R = a+(d|b)ab

adab --> (2, 10)

# RPQ Example

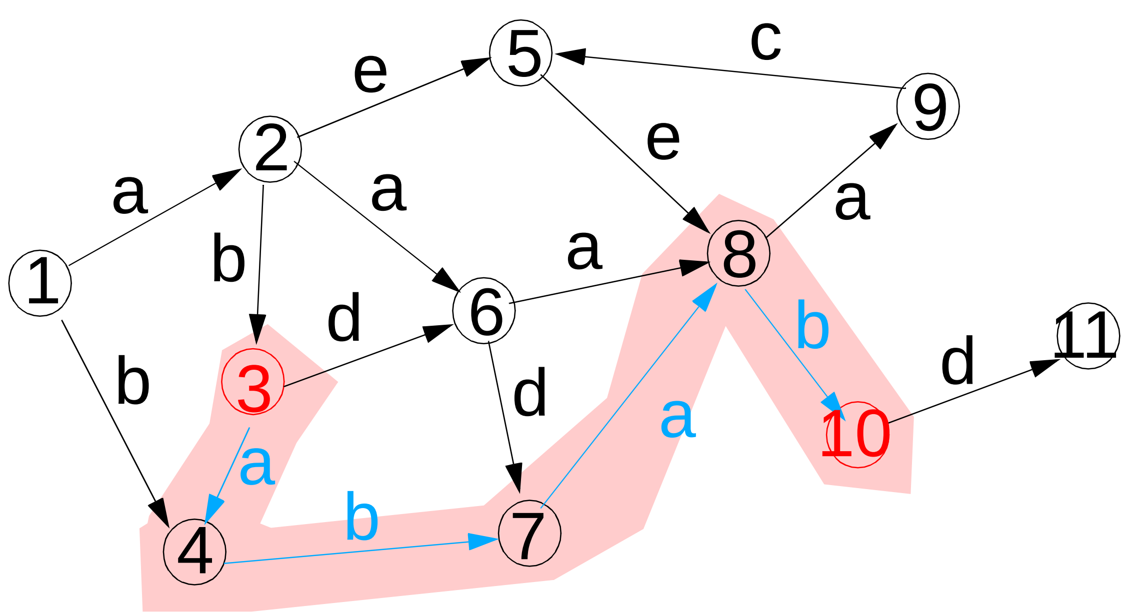



R = a+(d|b)ab

adab --> (2, 10)

aadab --> (1, 10)

# RPQ Example




R = a+(d|b)ab

adab --> (2, 10)

aadab --> (1, 10)

abab --> (3, 10)

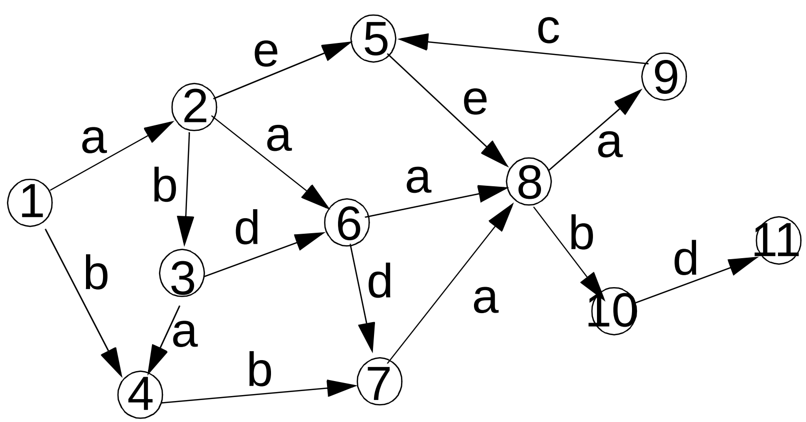
# Algorithms for Answering RPQ

- Mapping to finite automaton [Mendelzon, Wood, 1995]
  - Construct finite nondeterministic automata from query with start state $s_0$ und final state $s_f$
  - Consider G as NFA with start state x and final state y
  - Form product automaton
  - Determine if there is a path form ($s_0$, x) to ($s_f$, y)

- Search for rare labels and start BFS [Koschmieder, 2012]
  - Look for mandatory rare labels in the query (concering the graph)
  - Use the nodes from the rare edge labels as starting points for a two-way search between endpoints and startpoints of the rare edges

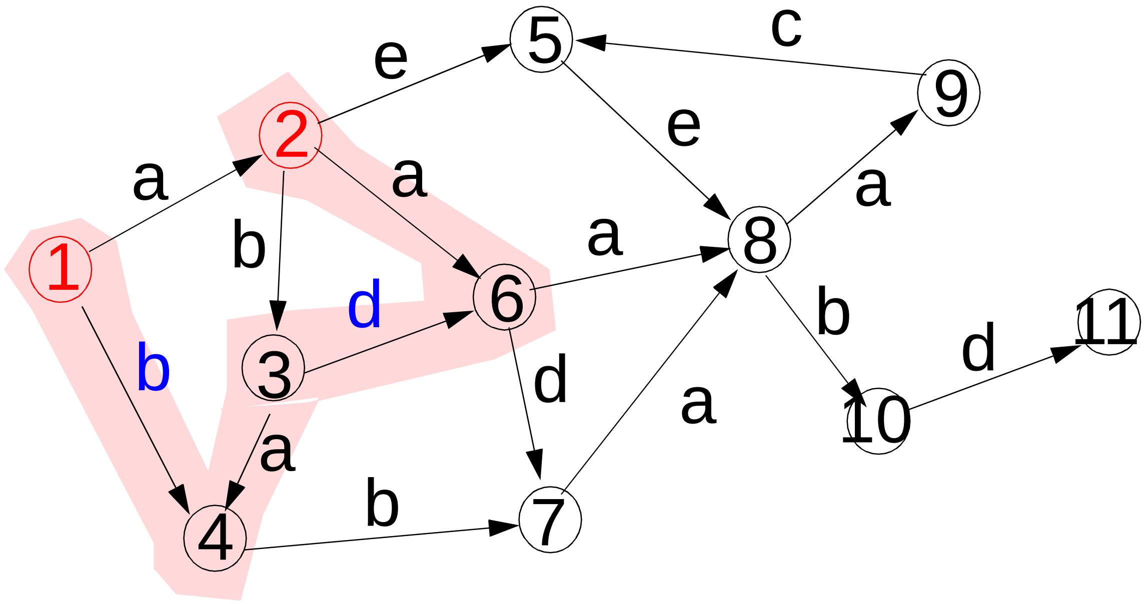
# Two-way Regular Path Queries (2RQP)

- 2RPQ extend the vocabulary of RPQ by the „inverse“ of each relationship symbol.
- Foreach symbol s in S: there exist a symbol $s^-$
- Example:




$R = a+d^-ab^-$

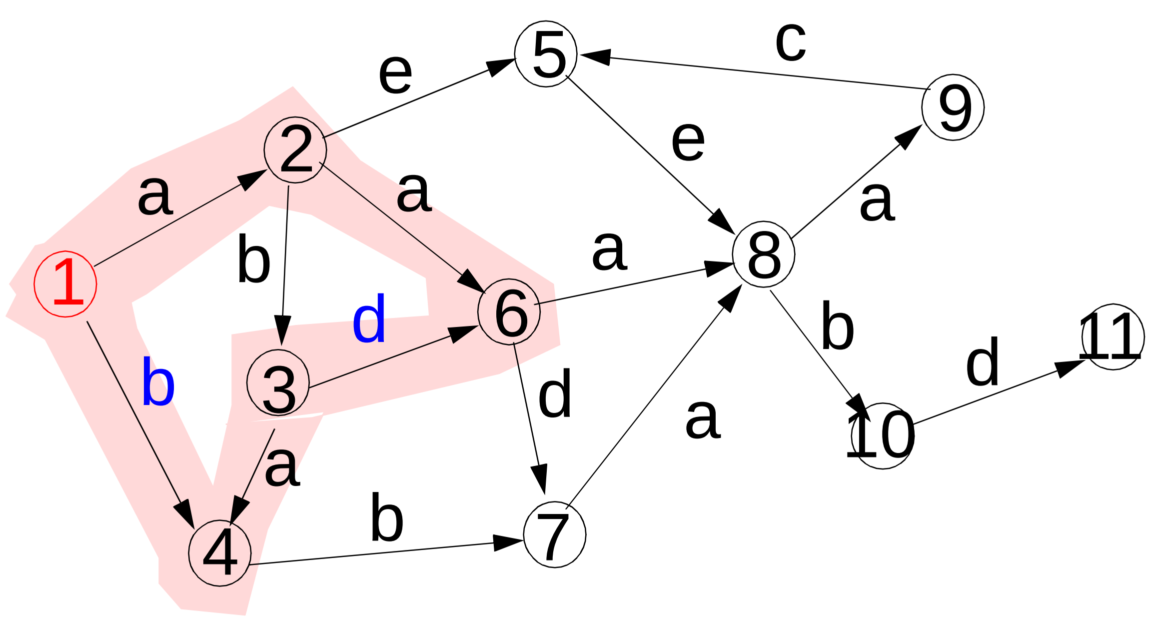
# 2RPQ Example




$R = a+d^{-}ab^{-}$

Results: (2,1)

# 2RPQ Example




$R = a+d^{-}ab^{-}$

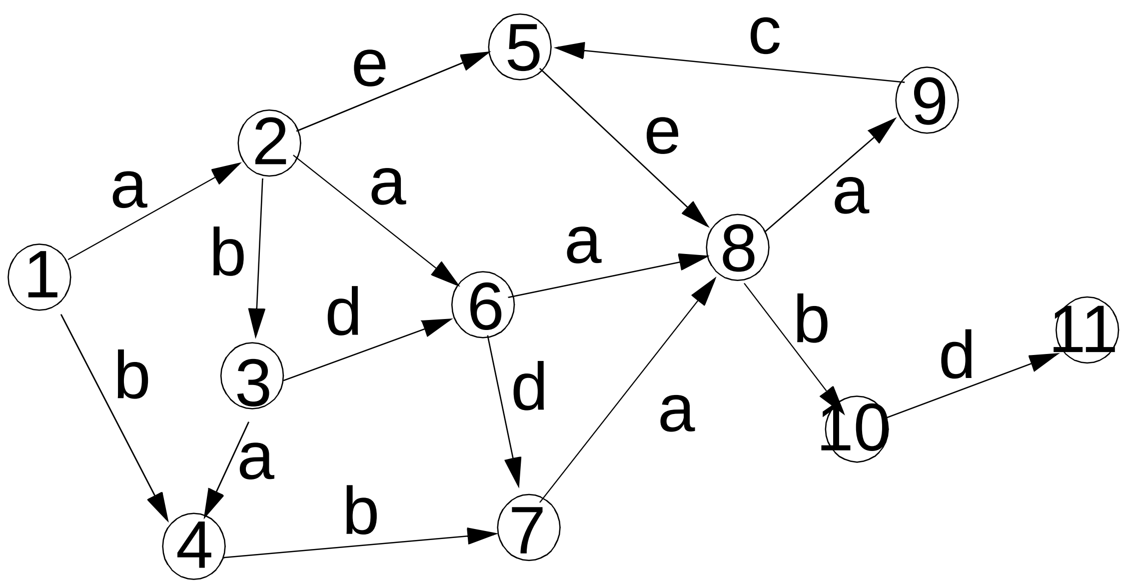
Results: (2,1)
(1,1)

# Conjunctive Regular Path Queries (CRPQ)

- $CRPQ(z_1, ..., z_n) = AND_{(1<=i<=m)} (x_i, R_i, y_i)$ // each $z_i$ is a $x_i$, $y_i$
- Examples:
  - Which couples are married by a pontifex?
    CRPQ(x,y,p) := (x, isMarriedWith, y) AND
    (x, isMarriedBy, p) AND
    (y, isMarriedBy, p) AND
    (p, isa, 'Pontifex')
  - Related at mostly over '5 edges' (navigating the family tree)
    CRPQ(x,y) := (x, isChildOf{5}, z) AND
    (y, isChildOf{5}, z)

# CRPQ Example

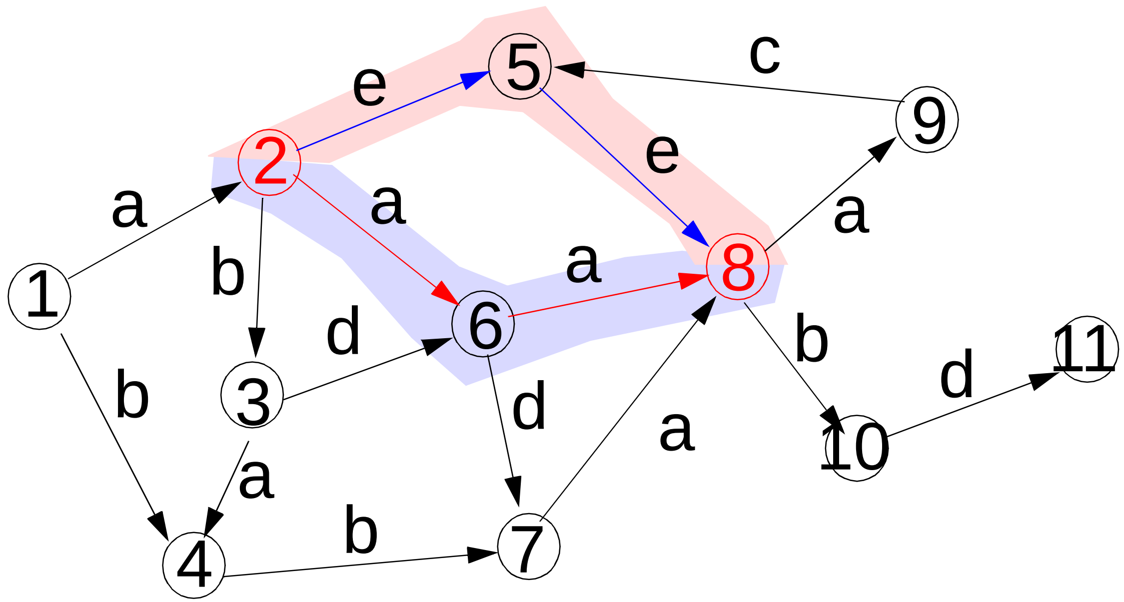
(x, a+, y)  AND  (x, e+, y)




.

# CRPQ Example

(x, a+, y)  AND  (x, e+, y)




Result: (2, 8)

# Extended Conjunctive Regular Path Queries (ECRPQ)

- CRPQ extended by
  - allow free path variables in the query
  - checking relations on sets of paths
- Example 1: Return all paths between x and y, which have a concrete node e (id:123) in between:
  - ECRPQ $(p_1, p_2)$ = $(x, R_1, e)$ AND $(e, R_2, y)$ AND (e, hasID, 123)
- Example 2: Path pattern match - Find all node connected by paths of the form $a^n b^n c^n$:
  - ECRPQ(x,y) = $(x, pv_1, z_1), (z_1, pv_2, z_2), (z_2, pv_3, y)$,
    $a+(pv_1), b+(pv_2), c+(pv_3)$,
    $el(pv_1, pv_2), el(pv_2, pv_3)$

$pv_3$ has the pattern c+

„equal-length“-relation

# Aggregation (AggCRPQ)

- CRPQ + Aggregation functions, i.e. for calculationg the distance between nodes
- Examples:
  - How many biological children does the husband of Carolyn has?:

    AggCRPQ (x,count(y)) = (x, isMarriedWith, ’Carolin’), (x, isParentOf, y)

  - Shortest path between x and y (with intermediate node z)

    AggCRPQ (x, y, min(len($p_1$)+len($p_2$))) = (x, $p_1$, z), (z, $p_2$, y)

# Summary & Outlook

- RPQ and its extensions are partly/complete realized in a number of graph query languages
- Different extensions of RPQ provide additonal power of expressiveness
- In most implementations of graph query languages RPQ are combined with additional data query functionalities
- Complexity and Containment is actual research field

# Literature


- Marcelo Fiore. Lecture Notes on Regular Languages and Finite Automata, Cambridge University Computer Laboratory, 2010
- Mendelzon, Wood. Finding regular simple path in graph databases. SIAM J. Computing., 24(6), 1995
- Peter Wood, Query Languages for Graph Databases; Sigmod Records (Volumne 41, No 1), 2012
- Pablo Barceló, Gaelle Fontaine; On the Data Complexity of Consistent Query Answering over Graph Databases. ICDT 2015.
- Pablo Barceló. Querying Graph Databases. PODS 2013.
- Pablo Barceló, Leonid Libkin, Carlos Hurtado, Peter Wood. Expressive languages for Path Queries over Graph-Structured Data, Pods 2010
- SPARQL Property Paths: http://www.w3.org/TR/sparql11-property-paths/

# SPARQL 1.1 path language

| Syntax Form | Matches |
| --- | --- |
| `uri` | A URI or a prefixed name. A path of length one. |
| `^elt` | Inverse path (object to subject). |
| `(elt)` | A group path `elt`, brackets control precedence. |
| `elt1 / elt2` | A sequence path of `elt1`, followed by `elt2` |
| `elt1 ^ elt2` | Shorthand for `elt1 / ^elt2`, that is `elt1` followed by the inverse of `elt2`. |
| `elt1 \| elt2` | A alternative path of `elt1`, or `elt2` (all possibilities are tried). |
| `elt*` | A path of zero or more occurrences of `elt`. |
| `elt+` | A path of one or more occurrences of `elt`. |
| `elt?` | A path of zero or one `elt`. |
| `elt{n,m}` | A path between n and m occurrences of `elt`. |
| `elt{n}` | Exactly `n` occurrences of `elt`. A fixed length path. |
| `elt{n,}` | `n` or more occurrences of `elt`. |
| `elt{,n}` | Between 0 and `n` occurrences of `elt`. |