Overview of Regular Path Queries in Graphs

Andreas Schmidt¹,², Iztok Savnik³

(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
- ...

Andreas Schmidt, Iztok Savnik - DBKDA - 2015
Types of Graphs

- Undirected
- Cyclic

Distances from http://www.luftlinie.org
Types of Graphs

- Directed
- Acyclic

Diagram:
- Country
  - Germany
  - City
    - Karlsruhe
    - Rom
    - Koper
    - Tokyo
Property Graph

- directed
- cyclic
- Nodes and relations have properties

name: France
population: 66.317.099
size: 668.763

hasCommonBorder

length: 448
orientation: East

name: Germany
population: 80.620.000
size: 357.340

hasCommonBorder

length: 448
orientation: West
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

![Diagram](image)
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:
    \[ \text{isChildOf}^+ \]
  - Cousins
    \[ \text{isChildOf}.\text{isMarriedWith}.\text{isChildOf} . \text{hasChild} . \text{hasChild} \]
RPQ Example

\[ R = a + (d|b)ab \]
RPQ Example

\[ R = a + (d | b)ab \]

adab --> (2, 10)
RPQ Example

\[ R = a + (d|b)ab \]

- \( \text{adab} \rightarrow (2, 10) \)
- \( \text{aadab} \rightarrow (1, 10) \)
RPQ Example

\[ R = a + (d|b)ab \]

- \( adab \rightarrow (2, 10) \)
- \( aadab \rightarrow (1, 10) \)
- \( abab \rightarrow (3, 10) \)
Algorithms for Answering RPQ

- Mapping to finite automaton [Mendelzon, Wood, 1995]
  - Construct finite nondeterministic automata from query with start state $s_o$ and 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 from $(s_o, x)$ to $(s_f, y)$

- Search for rare labels and start BFS [Koschmieder, 2012]
  - Look for mandatory rare labels in the query (concerning the graph)
  - Use the nodes from the rare edge labels as starting points for a two-way search between endpoints and start points of the rare edges
Two-way Regular Path Queries (2RQP)

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

\[ R = a + d^*ab^* \]
2RPQ Example

\[ R = a + d^{-1}ab^{-1} \]

Results: (2,1)
2RPQ Example

\[ R = a + d^\cdot a b^- \]

Results: (2,1)  
(1,1)
Conjunctive Regular Path Queries (CRPQ)

CRPQ(z₁, ..., zₙ) = \text{AND} \big( 1 \leq i \leq m \big) (xᵢ, Rᵢ, yᵢ)  // each zᵢ is a xᵢ, yᵢ

Examples:
- Which couples are married by a pontifex?
  CRPQ(x,y,p) := (x, isMarriedWith, y) \text{ AND}
  (x, isMarriedBy, p) \text{ AND}
  (y, isMarriedBy, p) \text{ AND}
  (p, isa, 'Pontifex')

- Related at mostly over '5 edges' (navigating the family tree)
  CRPQ(x,y) := (x, isChildOf{5}, z) \text{ AND}
  (y, isChildOf{5}, z)
CRPQ Example

\[(x, a+, y) \text{ AND } (x, e+, y)\]
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 \((p_1, p_2) = (x, R_1, e) \land (e, R_2, y) \land (e, \text{hasID}, 123)\)
- 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

„equal-length“-relation
Aggregation (AggCRPQ)

- CRPQ + Aggregation functions, i.e. for calculating the distance between nodes
- Examples:
  - How many biological children does the husband of Carolyn have?
    \[
    \text{AggCRPQ} (x, \text{count}(y)) = (x, \text{isMarriedWith}, 'Carolin'), (x, \text{isParentOf}, y)
    \]
  - Shortest path between \(x\) and \(y\) (with intermediate node \(z\))
    \[
    \text{AggCRPQ} (x, y, \text{min}(\text{len}(p_1)+\text{len}(p_2))) = (x, p_1, z), (z, p_2, y)
    \]
Summary & Outlook

• RPQ and its extensions are partly/complete realized in a number of graph query languages

• Different extensions of RPQ provide additional 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
- Peter Wood, Query Languages for Graph Databases; Sigmod Records (Volume 41, No 1), 2012
- Pablo Barceló, Gaëlle 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

<table>
<thead>
<tr>
<th>Syntax Form</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>A URI or a prefixed name. A path of length one.</td>
</tr>
<tr>
<td>^elt</td>
<td>Inverse path (object to subject).</td>
</tr>
<tr>
<td>(elt)</td>
<td>A group path elt, brackets control precedence.</td>
</tr>
<tr>
<td>elt1 / elt2</td>
<td>A sequence path of elt1, followed by elt2</td>
</tr>
<tr>
<td>elt1 ^ elt2</td>
<td>Shorthand for elt1 / ^elt2, that is elt1 followed by the inverse of elt2.</td>
</tr>
<tr>
<td>elt1</td>
<td>elt2</td>
</tr>
<tr>
<td>elt*</td>
<td>A path of zero or more occurrences of elt.</td>
</tr>
<tr>
<td>elt+</td>
<td>A path of one or more occurrences of elt.</td>
</tr>
<tr>
<td>elt?</td>
<td>A path of zero or one elt.</td>
</tr>
<tr>
<td>elt{n,m}</td>
<td>A path between n and m occurrences of elt.</td>
</tr>
<tr>
<td>elt{n}</td>
<td>Exactly n occurrences of elt. A fixed length path.</td>
</tr>
<tr>
<td>elt{n,}</td>
<td>n or more occurrences of elt.</td>
</tr>
<tr>
<td>elt{,n}</td>
<td>Between 0 and n occurrences of elt.</td>
</tr>
</tbody>
</table>