Big Graph Processing Systems

Part I: Graph Query Paradigms and their Semantics

▶ **Chapter 2:** Graph Query Language Classes

Christopher Spinrath

CNRS – LIRIS – Lyon 1 Université

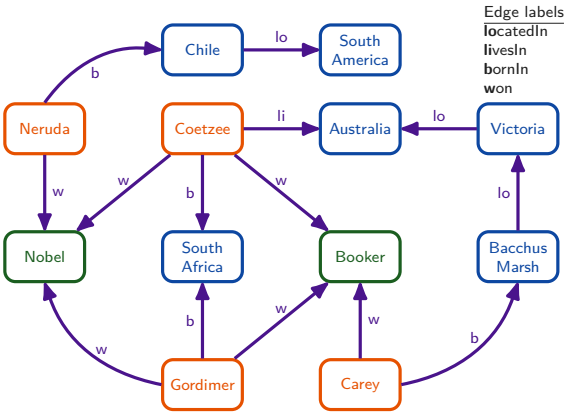
DISS Master 2025

This presentation is an adaption of slides from Angela Bonifati

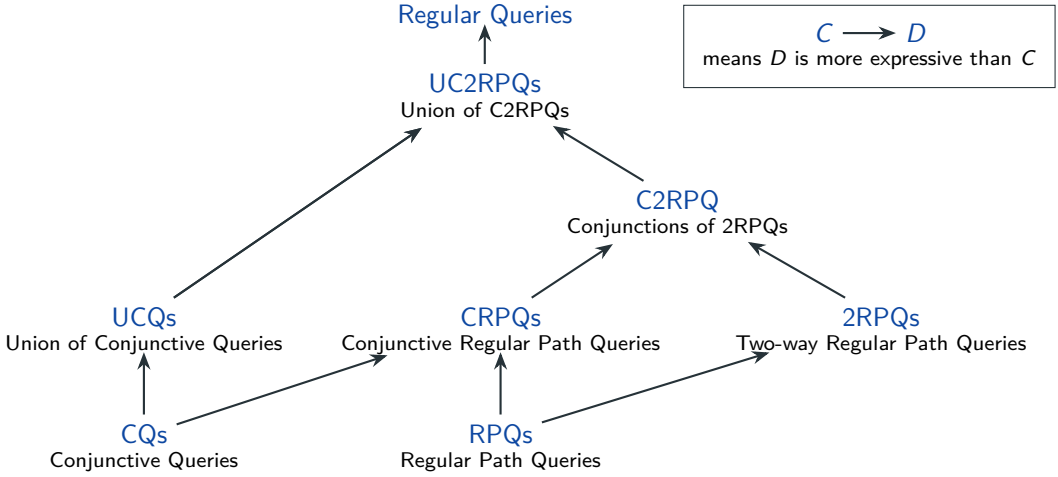
Graph Data Model

Graph data model for this chapter

- RDF-like data
- Directed graph with labeled edges
- $G = (V, E, \Sigma)$ where
  - V is the set of vertices,
  - $E \subseteq V \times \Sigma \times V$ is the set of labeled edges, and
  - Σ is the set (or alphabet) of labels.
- Node colors are for readability only



Query Language Classes – Overview



Conjunctive Queries (CQs) – The Idea

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

- Querying for patterns
- A query is given as as set of edge predicates (x_i, a_i, y_i)
- x_i, y_i are vertex variables or constants and a_i is an edge label

Example

- $(x, \text{hasWon}, \text{Nobel})$
- $(x, \text{hasWon}, \text{Booker})$
- $(x, \text{bornIn}, \text{South Africa})$

Conjunctive Queries (CQs) – The Idea

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

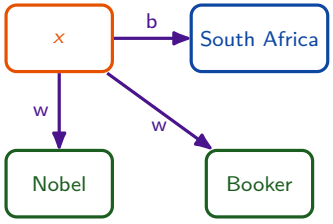
Idea

- Querying for patterns
- A query is given as as set of edge predicates (x_i, a_i, y_i)
- x_i, y_i are vertex variables or constants and a_i is an edge label

Example

- $(x, \text{hasWon}, \text{Nobel})$
- $(x, \text{hasWon}, \text{Booker})$
- $(x, \text{bornIn}, \text{South Africa})$

Visually



Conjunctive Queries (CQs)

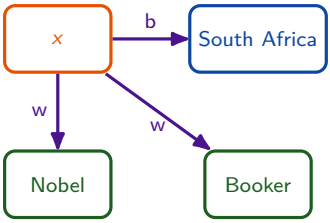
Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

- Each x_i and each y_i is a vertex variable or a constant
- Each $a_i \in \Sigma$ is an edge label
- Each *free variable* is some x_i or y_i

Visually



Conjunctive Queries (CQs)

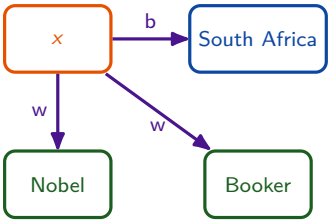
Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

- Each x_i and each y_i is a vertex variable or a constant
- Each $a_i \in \Sigma$ is an edge label
- Each *free variable* is some x_i or y_i
- A conjunctive query is a formula in the $\exists, \wedge$-fragment of first-order logic

Visually



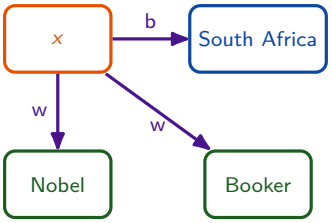
Conjunctive Queries (CQs)

Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

Visually



Semantics

- Let $\sigma \colon X \to V$ be a variable binding, i.e. a mapping of variables to vertices of the graph G

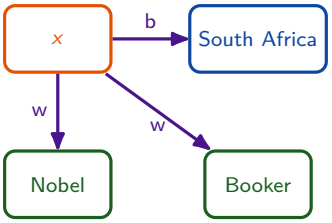
Conjunctive Queries (CQs)

Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

Visually



Semantics

- Let $\sigma \colon X \to V$ be a variable binding, i.e. a mapping of variables to vertices of the graph G
- Say relation $(G, \sigma) \models Q$ holds iff $\big(\sigma(x_i), a_i, \sigma(y_i)\big) \in E$ for all $1 \leq i \leq m$

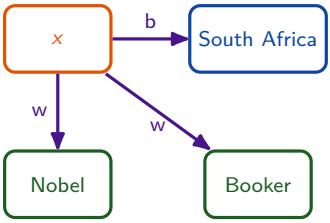
Conjunctive Queries (CQs)

Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, a_i, y_i)$$

Visually



Semantics

- Let $\sigma \colon X \to V$ be a variable binding, i.e. a mapping of variables to vertices of the graph G
- Say relation $(G, \sigma) \models Q$ holds iff $\big(\sigma(x_i), a_i, \sigma(y_i)\big) \in E$ for all $1 \leq i \leq m$
- Then the *query result* $Q(G)$ is the set of tuples $\big(\sigma(z_1), \ldots, \sigma(z_n)\big)$ such that $(G, \sigma) \models Q$:

$$Q(G) = \big\{ \big(\sigma(z_1), \ldots, \sigma(z_n)\big) \mid (G, \sigma) \models Q \big\}$$

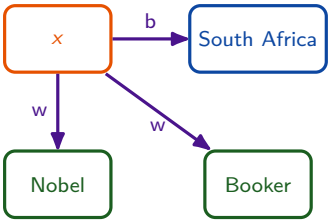
Conjunctive Queries (CQs)

Syntax

- A *conjunctive query* Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \le i \le m} (x_i, a_i, y_i)$$

Visually



Semantics

- Let $\sigma \colon X \to V$ be a variable binding, i.e. a mapping of variables to vertices of the graph G
- Say relation $(G, \sigma) \models Q$ holds iff $\big(\sigma(x_i), a_i, \sigma(y_i)\big) \in E$ for all $1 \le i \le m$
- Then the *query result* $Q(G)$ is the set of tuples $\big(\sigma(z_1), \ldots, \sigma(z_n)\big)$ such that $(G, \sigma) \models Q$:

$$Q(G) = \big\{ \big(\sigma(z_1), \ldots, \sigma(z_n)\big) \mid (G, \sigma) \models Q \big\}$$

Corresponds to a homomorphism

Conjunctive Queries (CQs) – Example

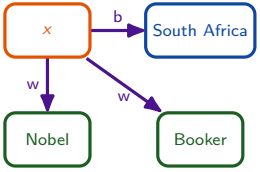
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

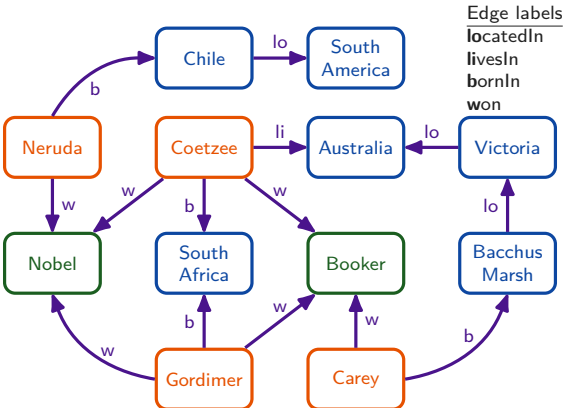
Example

All authors born in South Africa who have won both the Nobel and Booker prizes

$$\begin{aligned} \text{ans}(x) \leftarrow\ & (x, \text{hasWon}, \text{Nobel}) \\ & \wedge\ (x, \text{hasWon}, \text{Booker}) \\ & \wedge\ (x, \text{bornIn}, \text{South Africa}) \end{aligned}$$

Visually





Result?

Conjunctive Queries (CQs) – Example

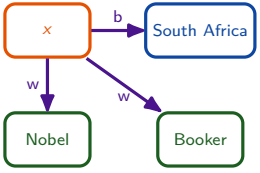
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

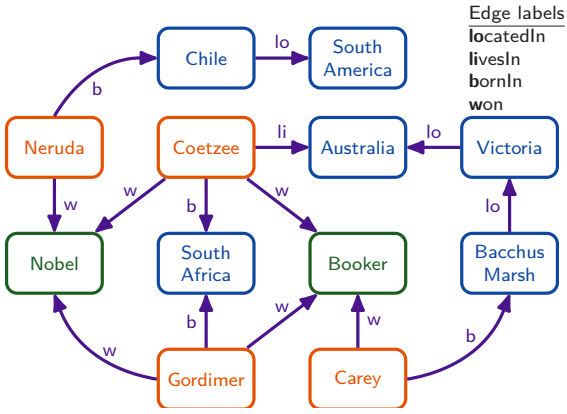
Example

All authors born in South Africa who have won both the Nobel and Booker prizes

$$\begin{aligned} \text{ans}(x) \leftarrow\ & (x, \text{hasWon}, \text{Nobel}) \\ & \wedge\ (x, \text{hasWon}, \text{Booker}) \\ & \wedge\ (x, \text{bornIn}, \text{South Africa}) \end{aligned}$$

Visually





Result?

Coetzee

Gordimer

Conjunctive Queries (CQs) – Example

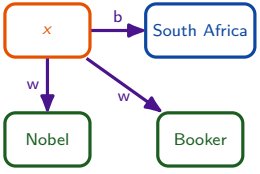
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

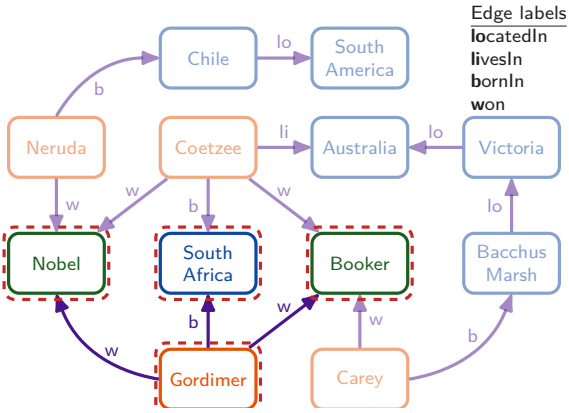
Example

All authors born in South Africa who have won both the Nobel and Booker prizes

$$\begin{aligned} \text{ans}(x) \leftarrow & (x, \text{hasWon}, \text{Nobel}) \\ & \wedge (x, \text{hasWon}, \text{Booker}) \\ & \wedge (x, \text{bornIn}, \text{South Africa}) \end{aligned}$$

Visually





Result?

Coetzee

Gordimer

Union of Conjunctive Queries (UCQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

Union of conjunctive queries with the same free variables

Union of Conjunctive Queries (UCQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

Union of conjunctive queries with the same free variables

Definition

A union of conjunctive queries Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigcup_{1 \leq j \leq k} Q_j(z_1, \ldots, z_n)$$

where each Q_j is a conjunctive query with free variables $z_1, \ldots, z_n$

Union of Conjunctive Queries (UCQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

Union of conjunctive queries with the same free variables

Definition

A union of conjunctive queries Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigcup_{1 \leq j \leq k} Q_j(z_1, \ldots, z_n)$$

where each Q_j is a conjunctive query with free variables $z_1, \ldots, z_n$

Semantics

The query result $Q(G)$ is the union $Q(G) = \bigcup_{1 \leq j \leq k} Q_j(G)$

Regular Path Queries (RPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

- Querying for reachability
- A query is given as a path predicate consisting of
  - a pair of vertex variables x, y, and
  - a path expression r
- A path expression is a regular expression over edge labels
- A pair of vertices is in the query result if and only if they are connected by a path conforming to the path expression

Example

$\text{ans}(x, y) \leftarrow (x, (\text{bornIn}|\text{livesIn}) \cdot \text{loctatedIn}^{\star}, y)$

Regular Path Queries (RPQs) – Syntax

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Syntax

- A regular path query Q is an expression

$$\text{ans}(x, y) \leftarrow (x, r, y)$$

- where x and y are vertex variables, and
- r is a regular expression over the alphabet Σ of edge labels
- Regular expressions over Σ are all and only those expressions recursively generated as follows:

Regular Path Queries (RPQs) – Syntax

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Syntax

- A regular path query Q is an expression

  $$\text{ans}(x, y) \leftarrow (x, r, y)$$

- where x and y are vertex variables, and
- r is a regular expression over the alphabet Σ of edge labels
- Regular expressions over Σ are all and only those expressions recursively generated as follows:
  - If $a \in \Sigma$, then a is a regular expression
  - If r_1 and r_2 are regular expressions then
    - $r_1 \cdot r_2$ (concatenation),
    - $r_1 | r_2$ (disjunction), and
    - $(r_1)^\star$ (Kleene star)

    are regular expressions

Regular Path Queries (RPQs) – Syntax

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Syntax

- A regular path query Q is an expression

  $$\text{ans}(x, y) \leftarrow (x, r, y)$$

- where x and y are vertex variables, and
- r is a regular expression over the alphabet Σ of edge labels
- Regular expressions over Σ are all and only those expressions recursively generated as follows:
  - If $a \in \Sigma$, then a is a regular expression
  - If r_1 and r_2 are regular expressions then
    - $r_1 \cdot r_2$ (concatenation),
    - $r_1 | r_2$ (disjunction), and
    - $(r_1)^*$ (Kleene star)

    are regular expressions
- Concatenation takes precedence, unnecessary parentheses are omitted

Regular Path Queries (RPQs) – Semantics

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Semantics

- Given a RPQ $Q = \text{ans}(x, y) \leftarrow (x, r, y)$ and a data graph G
- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:

Regular Path Queries (RPQs) – Semantics

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Semantics

- Given a RPQ $Q = \text{ans}(x, y) \leftarrow (x, r, y)$ and a data graph G
- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:
  - If $r = a \in \Sigma$, then $r(G) = \{(s, t) \mid (s, a, t) \in E\}$

Regular Path Queries (RPQs) – Semantics

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Semantics

- Given a RPQ $Q = \text{ans}(x, y) \leftarrow (x, r, y)$ and a data graph G
- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:
  - If $r = a \in \Sigma$, then $r(G) = \{(s, t) \mid (s, a, t) \in E\}$
  - If $r = r_1 \cdot r_2$, then

$$r(G) = \{(s, t) \mid \text{ there is } z \in V \text{ with } (s, z) \in r_1(G) \text{ and } (z, t) \in r_2(G)\}$$

Regular Path Queries (RPQs) – Semantics

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Semantics

- Given a RPQ $Q = \text{ans}(x, y) \leftarrow (x, r, y)$ and a data graph G
- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:
  - If $r = a \in \Sigma$, then $r(G) = \{(s, t) \mid (s, a, t) \in E\}$
  - If $r = r_1 \cdot r_2$, then

$$r(G) = \{(s, t) \mid \text{ there is } z \in V \text{ with } (s, z) \in r_1(G) \text{ and } (z, t) \in r_2(G)\}$$

  - If $r = r_1 | r_2$, then $r(G) = r_1(G) \cup r_2(G)$

Regular Path Queries (RPQs) – Semantics

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Semantics

- Given a RPQ $Q = \text{ans}(x, y) \leftarrow (x, r, y)$ and a data graph G
- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:
  - If $r = a \in \Sigma$, then $r(G) = \{(s, t) \mid (s, a, t) \in E\}$
  - If $r = r_1 \cdot r_2$, then

$$r(G) = \{(s, t) \mid \text{ there is } z \in V \text{ with } (s, z) \in r_1(G) \text{ and } (z, t) \in r_2(G)\}$$

  - If $r = r_1 | r_2$, then $r(G) = r_1(G) \cup r_2(G)$
  - If $r = (r_1)^\star$, then

$$r(G) = \text{TC}(r_1(G)) \cup \{(s, s) \mid s \in V\}$$

where $\text{TC}(R)$ denotes the transitive closure of binary relation R

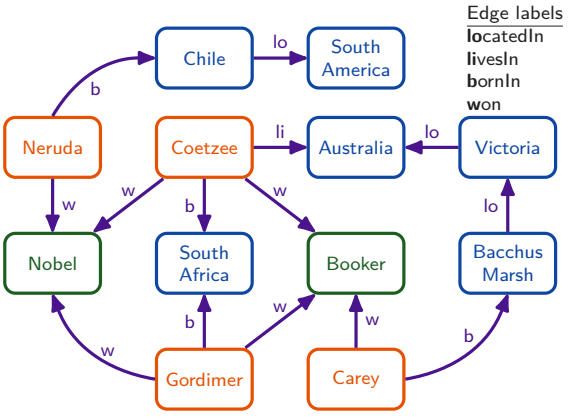
Regular Path Queries (RPQs) – Example

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Example

All authors and where they live in or are born

$$\text{ans}(x, y) \leftarrow (x, (\text{b}|\text{li}) \cdot \text{lo}^{\star}, y)$$



Regular Path Queries (RPQs) – Example

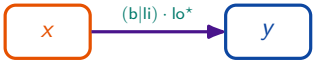
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

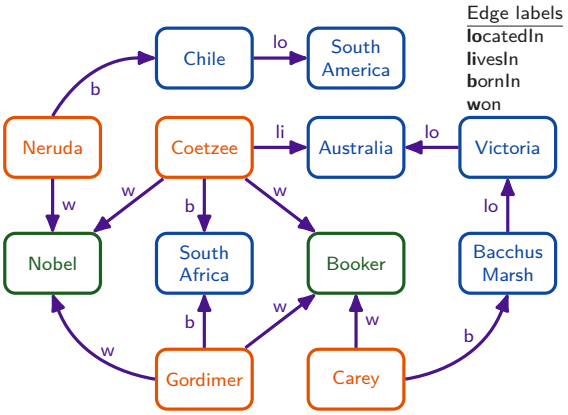
Example

All authors and where they live in or are born

$$\text{ans}(x, y) \leftarrow (x, (\text{b}|\text{li}) \cdot \text{lo}^\star, y)$$

Visually





Regular Path Queries (RPQs) – Example

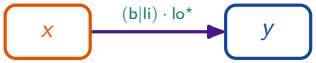
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Example

All authors and where they live in or are born

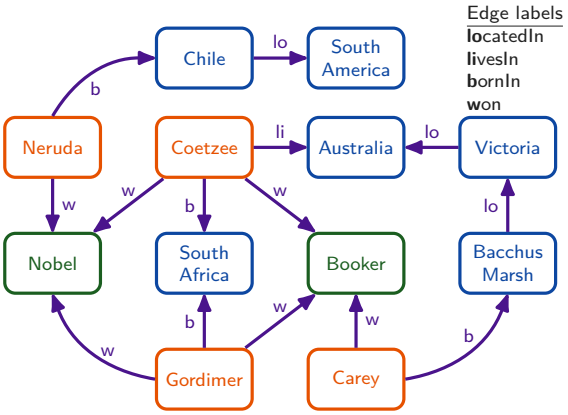
$$\text{ans}(x, y) \leftarrow (x, (\text{b}|\text{li}) \cdot \text{lo}^\star, y)$$

Visually



How many results?

A: 2 B: 8 C: 3 D: 6



Regular Path Queries (RPQs) – Example

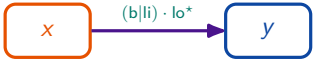
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Example

All authors and where they live in or are born

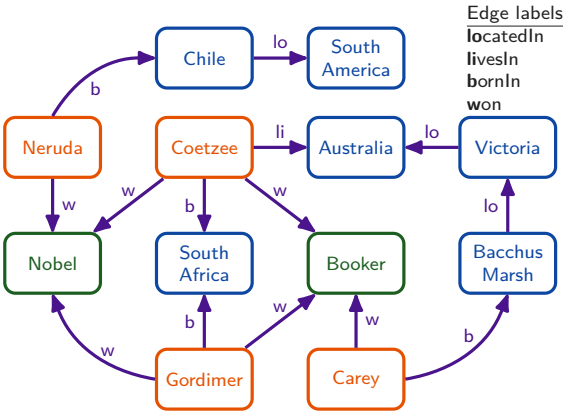
$$\text{ans}(x, y) \leftarrow (x, (\text{b}|\text{li}) \cdot \text{lo}^\star, y)$$

Visually



8 Results

Neruda	South America	Carey	Australia
Coetzee	Australia	Carey	Victoria
Coetzee	South Africa	Carey	Bacchus Marsh
Gordimer	South Africa	Neruda	Chile



Two-Way Regular Path Queries (2RPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Like RPQs but with the following extension

Syntax

- Regular expressions over Σ are all and only those expressions recursively generated as follows:

Two-Way Regular Path Queries (2RPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Like RPQs but with the following extension

Syntax

- Regular expressions over Σ are all and only those expressions recursively generated as follows:
  - If $a \in \Sigma$, then a is a regular expression
  - If $a \in \Sigma$, then a^{-1} is a regular expression (reverse traversal)
  - If r_1 and r_2 are regular expressions then $r_1 \cdot r_2$, $r_1|r_2$, and $(r_1)^\star$ are regular expressions

Two-Way Regular Path Queries (2RPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Like RPQs but with the following extension

Syntax

- Regular expressions over Σ are all and only those expressions recursively generated as follows:
  - If $a \in \Sigma$, then a is a regular expression
  - If $a \in \Sigma$, then a^{-1} is a regular expression (reverse traversal)
  - If r_1 and r_2 are regular expressions then $r_1 \cdot r_2$, $r_1|r_2$, and $(r_1)^\star$ are regular expressions

Semantics

- The query result $Q(G)$ of Q in G is the set $r(G)$ recursively defined as follows:
  - If $r = a \in \Sigma$, then $r(G) = \{(s,t) \mid (s,a,t) \in E\}$
  - If $r = a^{-1}, a \in \Sigma$, then $r(G) = \{(s,t) \mid (t,a,s) \in E\}$
  - If $r = r_1 \cdot r_2$, then $r(G) = \{(s,t) \mid$ there is $z \in V$ with $(s,z) \in r_1(G)$ and $(z,t) \in r_2(G)\}$
  - If $r = r_1|r_2$, then $r(G) = r_1(G) \cup r_2(G)$
  - If $r = (r_1)^\star$, then $r(G) = \mathrm{TC}(r_1(G)) \cup \{(s,s) \mid s \in V\}$ where $\mathrm{TC}(R)$ denotes the transitive closure of R

Conjunctive Regular Path Queries (CRPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

- Querying for patterns including variable length paths
- Conjunctions of RPQs

Conjunctive Regular Path Queries (CRPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

- Querying for patterns including variable length paths
- Conjunctions of RPQs

Syntax

- A conjunctive regular path query Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, r_i, y_i)$$

- The x_i, y_i are vertex variables or constants
- Each r_i is a regular expression over the label alphabet Σ

Conjunctive Regular Path Queries (CRPQs)

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Idea

- Querying for patterns including variable length paths
- Conjunctions of RPQs

Syntax

- A conjunctive regular path query Q is an expression

$$\text{ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, r_i, y_i)$$

- The x_i, y_i are vertex variables or constants
- Each r_i is a regular expression over the label alphabet Σ

Semantics

- Let $\sigma \colon X \to V$ be a variable binding
- Say relation $(G, \sigma) \models Q$ holds if and only if $(\sigma(x_i), \sigma(y_i)) \in r_i(G)$ for all $1 \leq i \leq m$
- Then the query result $Q(G)$ is the set of all tuples $(\sigma(z_1), \ldots, \sigma(z_n))$ such that $(G, \sigma) \models Q$

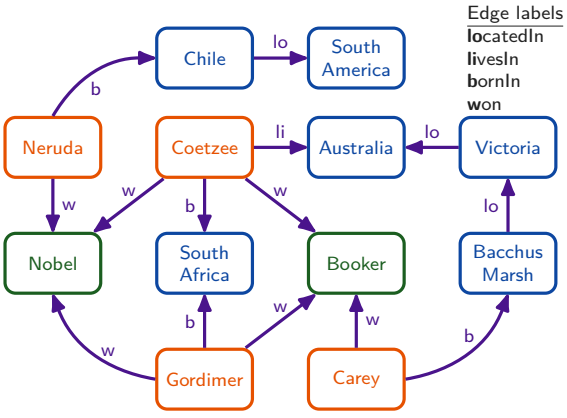
Conjunctive Regular Path Queries (CRPQs) – Example

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Example

All Nobel+Booker winners and where they live

$$\text{ans}(x, y) \leftarrow (x, \text{li} \cdot \text{lo}^\star, y)$$
$$\wedge\ (x, \text{w}, \text{Booker}) \wedge (x, \text{w}, \text{Nobel})$$



Conjunctive Regular Path Queries (CRPQs) – Example

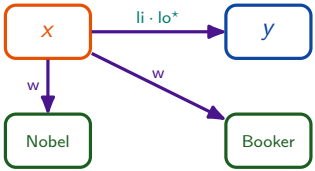
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

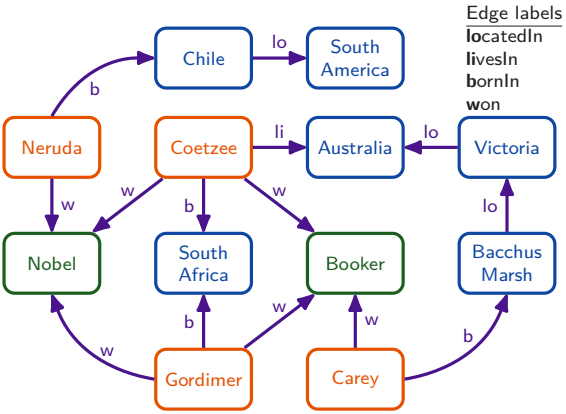
Example

All Nobel+Booker winners and where they live

$$\text{ans}(x, y) \leftarrow (x, \text{li} \cdot \text{lo}^\star, y) \\ \wedge (x, \text{w}, \text{Booker}) \wedge (x, \text{w}, \text{Nobel})$$

Visually





Conjunctive Regular Path Queries (CRPQs) – Example

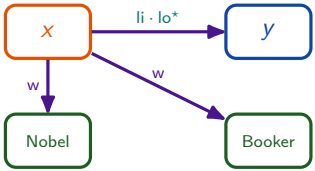
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

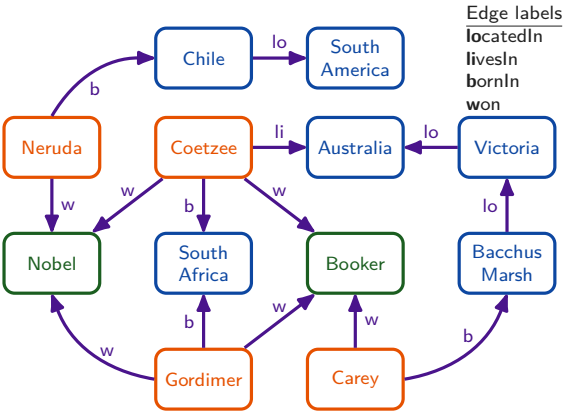
Example

All Nobel+Booker winners and where they live

$$\text{ans}(x, y) \leftarrow (x, \text{li} \cdot \text{lo}^\star, y)$$
$$\wedge\ (x, \text{w}, \text{Booker}) \wedge (x, \text{w}, \text{Nobel})$$

Visually





Result?

A: (Gordimer, Australia) B: (Coetzee, Australia) C: (Coetzee, Bacchus Ma.) D: (Neruda, South Am.)

Conjunctive Regular Path Queries (CRPQs) – Example

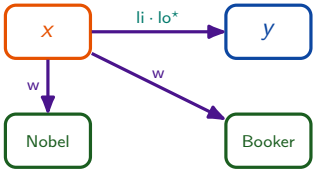
[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

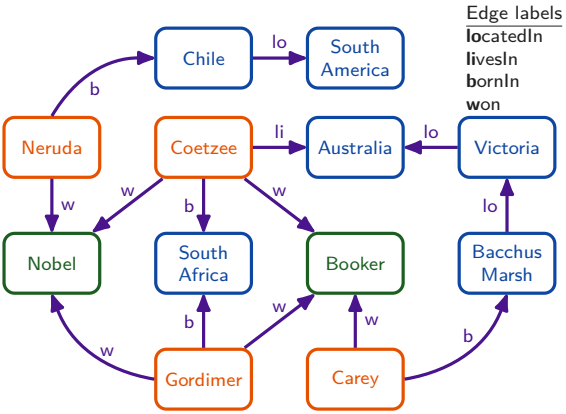
Example

All Nobel+Booker winners and where they live

$$\text{ans}(x, y) \leftarrow (x, \text{li} \cdot \text{lo}^\star, y)$$
$$\wedge\ (x, \text{w}, \text{Booker}) \wedge (x, \text{w}, \text{Nobel})$$

Visually





Result?

A: (Gordimer, Australia) B: (Coetzee, Australia) C: (Coetzee, Bacchus Ma.) D: (Neruda, South Am.)

C2RPQs and UC2RPQs

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

C2RPQs

- Conjunctions of 2RPQs

UC2RPQs

- Unions of C2RPQs
- Class which most practical query languages (more or less) fall into

Example

$$\begin{aligned} &\mathsf{ans}(x,y) \leftarrow (x, \mathsf{actedIn} \cdot \mathsf{actedIn}^{-1} \cdot (\mathsf{actedIn} \cdot \mathsf{actedIn}^{-1})^{\star}, y) \\ &\qquad\qquad \wedge (x, \mathsf{livesIn}, z) \wedge (y, \mathsf{livesIn}, z) \\ \cup\ &\mathsf{ans}(x,y) \leftarrow (x, \mathsf{worksFor} \cdot \mathsf{partOf}^{\star}, w) \wedge (y, \mathsf{worksFor} \cdot \mathsf{partOf}^{\star}, w) \end{aligned}$$

, Regular Queries

[Wood, "Query languages for graph databases", *SIGMOD Rec.*, 2012]

Regular Queries

- UC2RPQs can be nested, if they define a binary relation (that is, a set of edges)
- The left hand sides (heads) of other, binary UC2RPQs can be used as any other label
  - In particular, as part of regular expressions

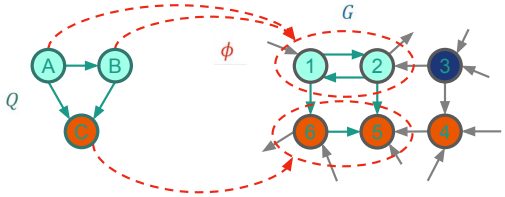
Example

$$\begin{aligned} \text{ans}(x, c_x, y, c_y) \leftarrow (x, \text{livesIn}, c_x) \wedge (x, \text{livesIn}, c_x) \\ \wedge (c_x, \text{partnerCities}, c_y) \wedge (x, \text{friendsWithMutualFriend}^\star, y) \\ \text{friendsWithMutualFriend}(x, y) \leftarrow (x, \text{friend}, y) \wedge (x, \text{friend}, z) \wedge (y, \text{friend}, z) \end{aligned}$$

Summary

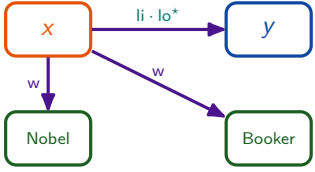
Matching Semantics

- Graph simulation
- Dual simulation
- Subgraph isomorphism
- Subgraph homomorphism

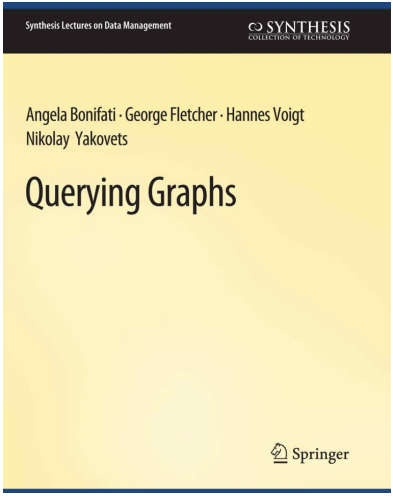


Query Types

- Conjunctive Queries (CQs)
- Regular Path Queries (RPQs)
- Conjunctive Regular Path Queries (CRPQs)
- Union of Conjunctive Two-Way Regular Path Queries (UC2RPQs)
- Regular Queries



Literature



Bonifati et al., *Querying Graphs*, 2018

References


Bonifati, Angela, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets (2018). *Querying Graphs*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers. DOI: 10.2200/S00873ED1V01Y201808DTM051. URL: https://doi.org/10.2200/S00873ED1V01Y201808DTM051.


Wood, Peter T. (2012). "Query languages for graph databases". In: *SIGMOD Rec.* 41.1, pp. 50–60. DOI: 10.1145/2206869.2206879. URL: https://doi.org/10.1145/2206869.2206879.