Query languages & query evaluation

08/2021

REGULAR PATH QUERIES

An essential feature behind practical graph DB languages is the ability to "navigate" the data:

Traverse paths in the graph database while checking if a certain condition holds.

Given a graph database $G = (V, E)$, a path in G is a sequence:

$$\varphi = v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \xrightarrow{a_3} \cdots v_{n-1} \xrightarrow{a_n} v_n,$$

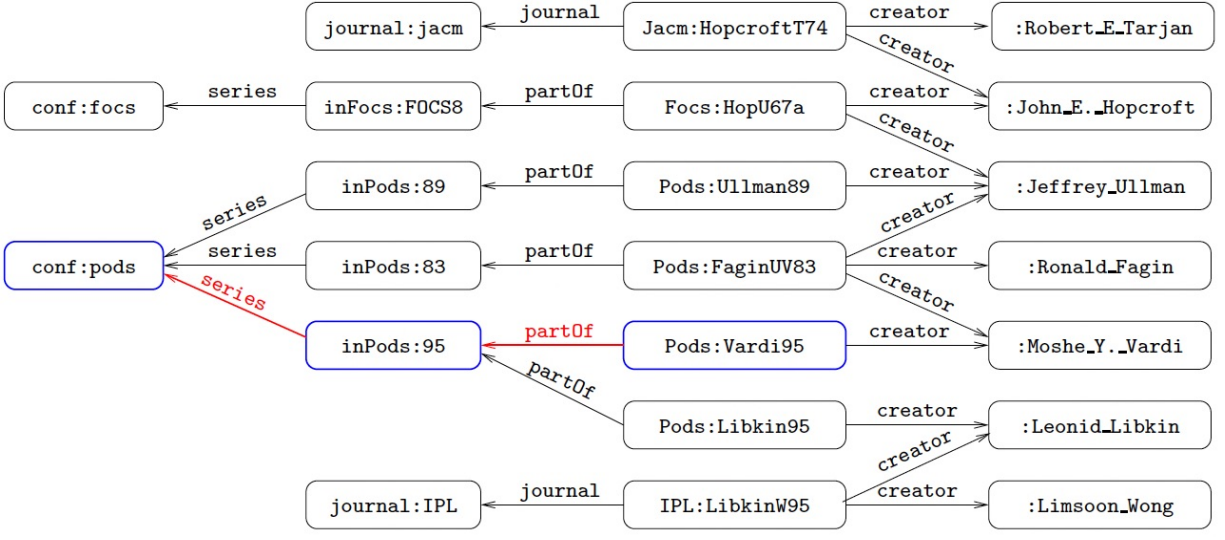
such that $(v_{i-1}, a_i, v_i) \in E$, for all $1 \leq i \leq n$.

The label of path φ is

$$lab(\varphi) = a_1 \ldots a_n \in \Sigma^*.$$

Observation: For each node $v \in V$, the sequence $\varphi = v$ is a path with $lab(\varphi) = \varepsilon$.

Example: Below we see a path in the graph DB highlighted in colors blue (nodes) and red (edges).



A regular path query (RPQ) over finite alphabet Σ is an expression

$$q := x \xrightarrow{L} y,$$

where L is a regular expression over Σ.

Example: The following are RPQs over $\Sigma = \{a, b\}$:

$$x \xrightarrow{(aa)^*} y$$

$$x \xrightarrow{(a+b)^* b} y.$$

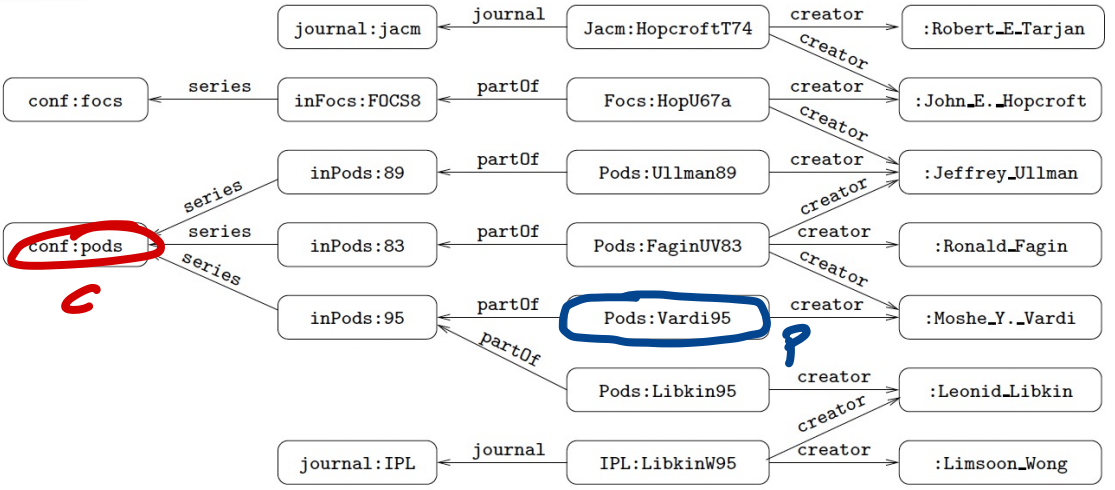
Semantics of RPQs: Given a graph database G and a RPQ $q := x \xrightarrow{L} y$ over Σ, the evaluation of q on G, denoted $q(G)$, is the set

$$\{(u,v) \in V \times V \mid \text{there exists a certain kind of path } \varphi \text{ from node } u \text{ to node } v \text{ in } G \text{ s.t. } lab(\varphi) \in L\}.$$

Different semantics based on the class of paths allowed:

- **Arbitrary path semantics**: Any path is considered to be valid.
- **Simple path semantics**: Based on paths without repeated nodes (simple paths).
- **Trail path semantics**: Based on paths without repeated edges (trails).

Example: The RPQ $x \xrightarrow{part_of \cdot series} y$ selects pairs (p, c), such that p is a paper published in conference c.



This under any of the previous semantics.

In general:

(u,v) linked by a single path labeled in L

$\Rightarrow$

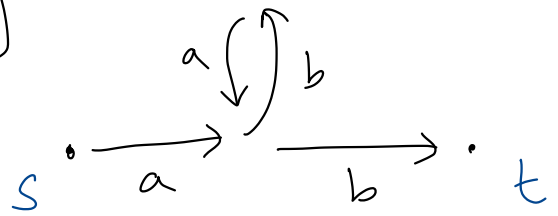
(u,v) linked by a trail Labeled in L

$\Rightarrow$

(u,v) linked by an arbitrary path labeled in L.

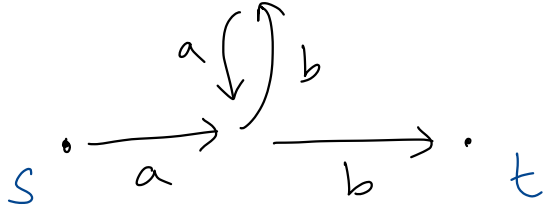
The converse statements are false:

1.



- there is a trail from s to t labed abab.
- there is no simple path from s to t labeled abab.

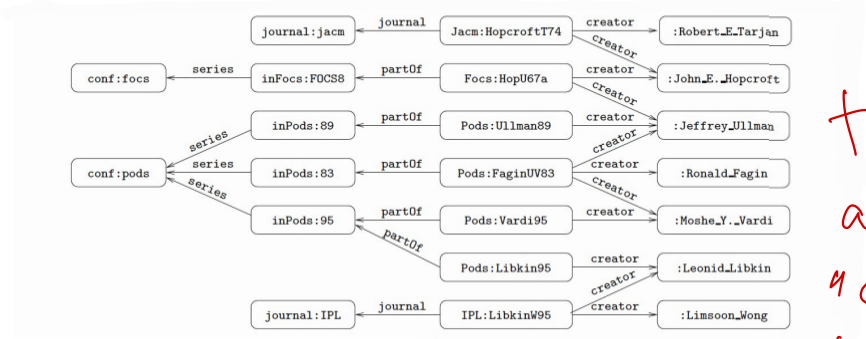
2



- there is a path from s to t labeled ababab.
- there is no trail from s to t labeled ababab

The need for inverses:

Imagine that we want to extract pairs (a, c) such that a is an author who published a paper in conference c.



We cannot do this with RPQs as edges labeled "creator" need to be traversed backwards.

RPQs with inverses (2RPQs):

Σ := finite alphabet

$\Sigma^{\pm} := \Sigma \cup \{a^- \mid a \in \Sigma\}$.

A **2RPQ** over Σ is a query

$$q := x \xrightarrow{L} y,$$

with L a regular expression over $\Sigma^{\pm}$.

Example: The following is a 2RPQ over Σ:

$$x \xrightarrow{aba^-b^-a} y$$

Semantics of 2RPQs:

$G = (V, E)$ graph database over Σ

$G^{\pm} = (V, E^{\pm})$ over $\Sigma^{\pm}$ such that

$$E^{\pm} = E \cup \{(v \xrightarrow{a^-} u) \mid (u \xrightarrow{a} v) \in E\}.$$

Given 2RPQ q over Σ, we define:

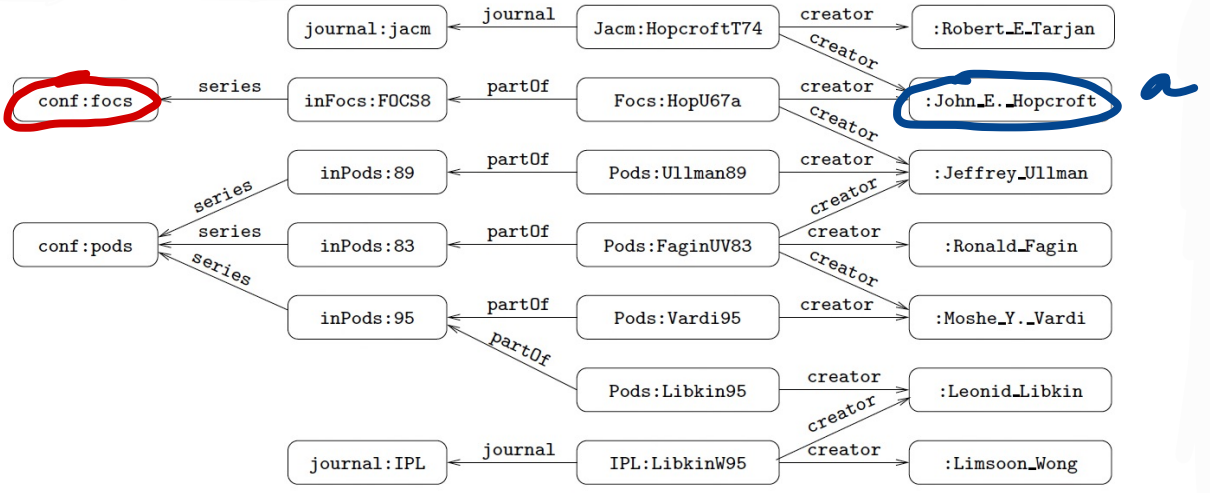
$$q(G) := q(G^{\pm}),$$

where $q(G^{\pm})$ is well-defined since q is an RPQ over $\Sigma^{\pm}$.

Example: The 2RPQ $x \xrightarrow{\text{creator} \cdot \text{partof} \cdot \text{series}} y$

selects pairs (a, c), such that

a is an author that published in conference c.



This under any of the previous semantics.

COMPLEXITY OF 2RPQ EVALUATION

PROBLEM: RPQ-evaluation

Input: Graph DB $G = (V, E)$, nodes $s, t \in V$, and RPQ $q := x \xrightarrow{L} y$

Output: YES, if $(s, t) \in q(G)$, i.e. there is path φ from s to t in G with $lab(\varphi) \in L$.

Bottomline: The complexity of RPQ-evaluation depends on the path semantics that we use.

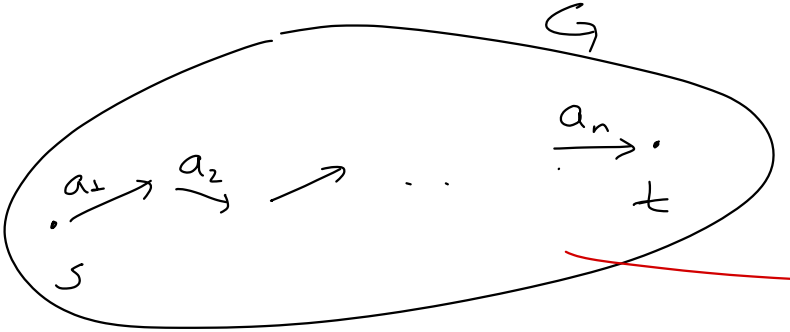
For distinguishing cases, we use:

- $q_{arb}(G)$: $\{(u,v) \in V \times V \mid$ there is arbitrary path φ from s to t with $lab(\varphi) \in L\}$

- $q_{simple}(G)$: Same for simple paths.
- $q_{trail}(G)$: Same for trails.

Arbitrary path semantics:

We want to check:



$a_1 \dots a_n \in L.$

$(s,t) \in q_{arb}(G).$

Idea:

- Turn L into automata A_L
- "See" G as an automata A_G.
- Take the intersection of both.

Formally:

1. Turn regular expression L over $\Sigma^{\pm}$ into an equivalent automaton $\mathcal{A}_L = (Q, q_0, F, \delta)$.
   This can be done in time $O(|L|)$.
2. From graph DB G and nodes s, t, compute in $O(|G|)$ automaton $\mathcal{A}_G^{s,t} = (V, s, \{t\}, \delta')$, with $\delta' = \{(u, a, v), (v, a^-, u) \mid u \xrightarrow{a} v \in E\}$.

$\Rightarrow$ words accepted by $\mathcal{A}_G$ are the labels of paths from s to t in G.

3. Construct automaton

$$A_G^{s,t} \times A_L = \Big(Q \times V, (q_0, s), \{(f,t) \mid f \in F\},$$
$$\delta'' = \big\{[(q,\mu), a, (q',\nu)] \;\big|\; (q,a,q') \in \delta \text{ and } (\mu,a,\nu) \in \delta'\big\}\Big)$$

This can be done in time

$$O(|A_G^{s,t}| \times |A_L|) = O(|G| \times |L|).$$

$\Rightarrow$ words accepted by $A_G^{s,t} \times A_L$ are those that label paths in G from s to t and belong to L

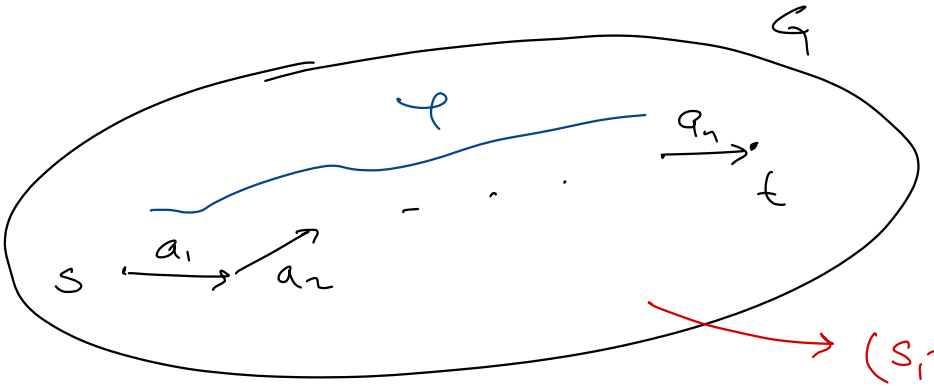
4. Check that $A_G^{s,t} \times A_L$ accepts <u>some</u> word

This can be done in time $O(|G| \cdot |L|)$.

Theorem: The problem RPQ evaluation under the arbitrary path semantics can be solved in linear time: $O(|G| \cdot |A|)$.

Simple Path semantics:

Given graph DB $G = (V, E)$, nodes s, t, we want to check:



- $a_1 \dots a_n \in L$
- φ does not have repeated nodes

$\rightarrow (s, t) \in q_{simple}(G)$.

Theorem (Mendelzon, Wood; '95)

RPQ-evaluation under the simple path semantics is NP-complete, even in data complexity.

Proof:

- Consider the RPQ $q := x \xrightarrow{(aa)^*} y$
- Under the simple path semantics, this checks if there is a simple path of even length in a graph.
- The latter is NP-complete (Lapaugh, Papadimitriou; '84).

Theorem (Bagan, Bonifati, Groz; 2013)

Assuming $\mathcal{P} \neq NP$, the following are equivalent for each RPQ $q = x \xrightarrow{L} y$, with L fixed:

a) q can be evaluated in $\mathcal{P}$ under simple path semantics

b) L can be expressed as a regular expression in class ψ_{tr} with:

$\psi_{seq} := \bar{\omega} + \varepsilon \mid A^{\geq k} + \varepsilon \mid \psi_{seq} \cdot \psi_{seq}$ $\left(\begin{array}{l}\bar{\omega} \in \Sigma'^{*} \\ A \subseteq \Sigma'\end{array}\right)$

$\psi_{tr} := \bar{\omega}\, \psi_{seq}\, \bar{\omega}' \mid \psi_{tr} + \psi_{tr}$

Example: Evaluation of RPQ

$q : x \xrightarrow{L} y$ in data complexity under simple path semantics is:

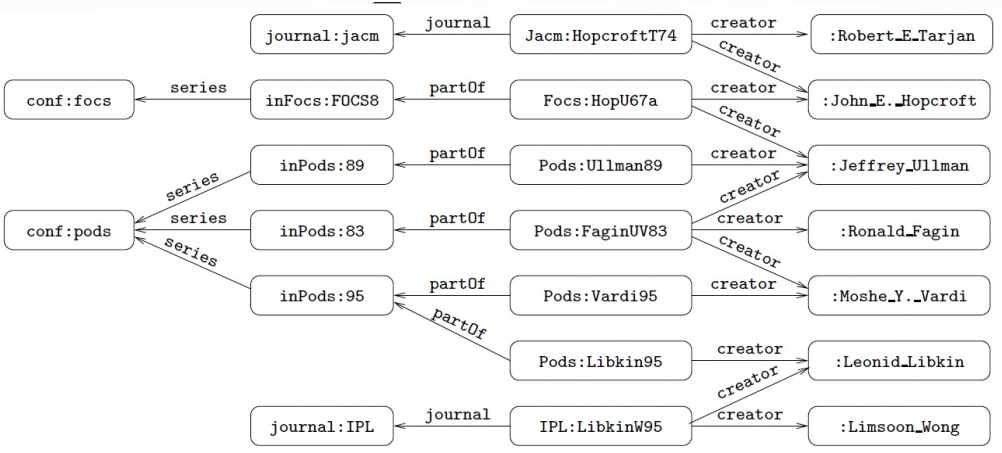
1. In $\mathcal{P}$, if L is a finite regular language.
2. In $\mathcal{P}$, for $L = a^*(bbb^* + \varepsilon)c^*$.
   $= (a^{\geq 1} + \varepsilon)(b^{\geq 2} + \varepsilon)(c^{\geq 1} + \varepsilon)$
3. Not in $\mathcal{P}$, assuming $\mathcal{P} \neq NP$, for $L = (aa)^*$.

Trail path semantics:

- As for the case of the simple path semantics, evaluation of RPQs under trails can be NP-complete (in data complexity).
- There is also a precise characterization of tractability for evaluation of RPQs under trail semantics [Martens, Niewerth, Trautner; 2019].

CONJUNCTIVE RPQs.

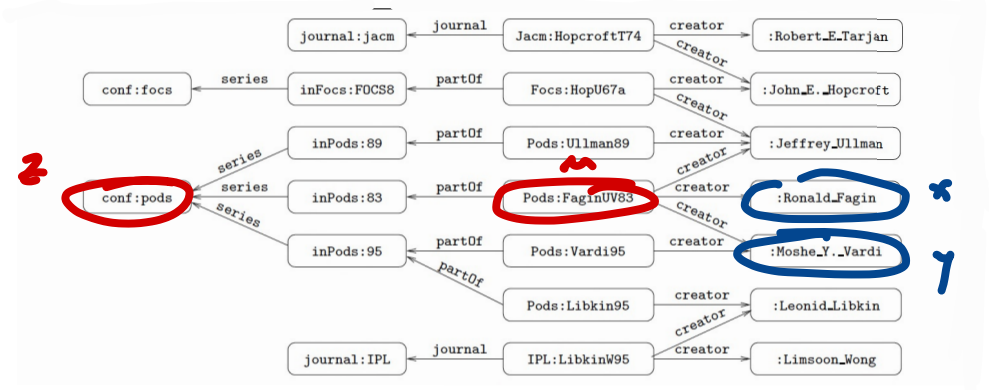
Example: Imagine we want to extract pairs of authors who have a joint paper in a conf.



This cannot be expressed with a 2RPQ!

Idea: Use the "conjunction" of three RPQs and then "project" on some variables:

$$Ans(x,y) :- (x \xrightarrow{creator} m), (y \xrightarrow{creator} m), (m \xrightarrow{partOf.series} z).$$



Only x, y are projected in the result!

A conjunctive 2RPQ (C2RPQ) over Σ is an expression of the form:

$$Ans(\bar{z}) :- (x_1 \xrightarrow{L_1} y_1), \ldots, (x_m \xrightarrow{L_m} y_m),$$

where:

- The x_i's and y_i's are variables;
- $\bar{z}$ is a tuple of variables in $\{x_1, \ldots, x_m, y_1, \ldots, y_m\}$
- Each L_i is a regular expression over $\Sigma^{\pm}$.

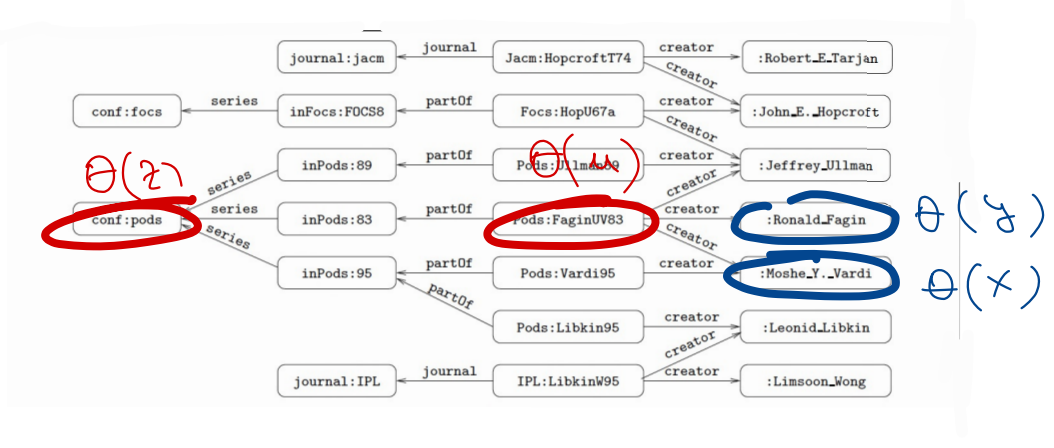
Given a graph DB $G = (V, E)$
and a CRPQ $q := Ans(\bar{z}) :- \bigwedge_{i=1}^{m} (x_i \xrightarrow{L_i} y_i)$,
a homomorphism from q to G
is a mapping $\theta : \{x_1, \dots, x_m, y_1, \dots, y_m\} \to V$
such that:

there is a path ρ from
$\theta(x_i)$ to $\theta(y_i)$ with $lab(\rho) \in L_i$,
for every $i \leq m$.

We assume an arbitrary path
semantics for simplicity.

Example: Below we see a homomorphism from the given C2RPQ into the graph database $G = (V, E)$.

$$Ans(x, y) :- (x \xrightarrow{creator} u), (y \xrightarrow{creator} u), (u \xrightarrow{partOf\text{-}series} z).$$



The evaluation of C2RPQ

$q := Ans(\bar{z}) :- \bigwedge_{i=1}^{m} (x_i \xrightarrow{L_i} y_i)$

on a graph database $G = (V, E)$ is the set: $q(G) = \{\theta(\bar{z}) \mid \theta \text{ is a homomorphism from } q \text{ to } G\}$.

Problem: C2RPQ-evaluation

Input: A graph database $G = (V, E)$, a C2RPQ $q := Ans(\bar{z}) :- \bigwedge_{i=1}^{m} (x_i \xrightarrow{L_i} y_i)$, a tuple $\bar{t}$ of nodes with $|\bar{t}| = |\bar{z}|$.

Output: YES, if $\bar{t} \in q(G)$.

Theorem: C2RPQ-evaluation is NP-complete.

Proof: We first show that C2RPQ-evaluation is in NP.

The non-deterministic algorithm is as follows:

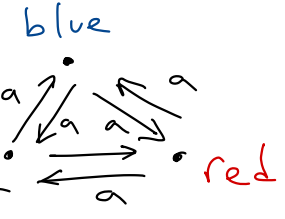
1. Guess a polynomially-sized mapping $\theta : \{x_1, \ldots, x_m, y_1, \ldots, y_m\} \to V$.
2. Check in polynomial time that $\forall i \leq m$ there is path φ_i from $\theta(x_i)$ to $\theta(y_i)$ with $\text{lab}(\varphi_i) \in L_i$.
3. Check in polynomial time that $\theta(\bar{z}) = \bar{t}$.

We now prove the NP-hardness:
We use a reduction from 3-colorability.

- Take an undirected graph
  $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$.
- From G, we build a CRPQ

$$q := \mathrm{Ans}() \leftarrow \bigwedge_{(v_i, v_j) \in E} (x_i \xrightarrow{a} x_j)$$

and a graph database $H :=$



over $\Sigma = \{a\}$ in polynomial time.

Claim: G is 3-colorable $\iff$ there is a homomorphism from q to H.

- In fact, a homomorphism from q to H must map adjacent nodes into different nodes / colors in $\{$blue, red, black$\}$.
- In turn, if $G = (V, E)$ is 3-colorable via $\eta : V \to \{$blue, red, black$\}$, then η is a homomorphism from q to H.

Corollary: C2RPQ - evaluation is NP-hard even if:

1. Queries use no inverses.
2. ALL edges are labeled a.
3. ALL regular expressions in queries are of the form $x \xrightarrow{a} y$.
4. The graph database $\mathcal{G}$ is fixed.

Intuition: The cost of evaluation comes from the "structure" of the C2RPQ [arbitrary graph].

Data Complexity of C2RPQ-evaluation

- In the previous reduction, the C2RPQ we construct is <u>not</u> fixed: it depends on the graph G.
- This does not tell us that C2RPQ-evaluation is NP-complete in data complexity.
- In fact, if q is fixed we can dramatically reduce the complexity of evaluation.

Assume that $q := Ans(\bar{z}) :- \bigwedge_{i=1}^{m} (x_i \xrightarrow{L_i} y_i)$ is fixed. So, given a graph DB $G = (V, E)$ and tuple $\bar{t}$ in V with $|\bar{t}| = |\bar{z}|$, we need to check if $\bar{t} \in q(G)$.

We apply the following:

1. List in polynomial time $|G|^{O(|q|)}$ all mappings $\theta : \{x_1, \ldots, x_m, y_1, \ldots, y_m\} \to V$.
2. For each such θ, check if θ is a homomorphism in polynomial time. Delete it otherwise.
3. Check if there is θ in the list with $\theta(\bar{z}) = \bar{t}$.

- Hence, C2RPQ-evaluation can be solved in $\mathcal{P}$ in data complexity.
- But, actually something stronger holds:

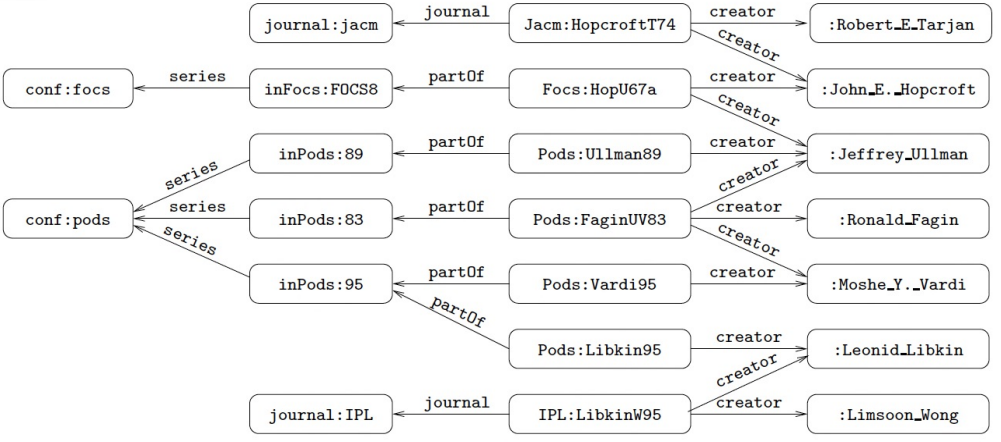
Theorem: C2RPQ-Evaluation can be solved in NLOGSPACE in data complexity.

Proof:

- Mappings $\theta : \{x_1, \dots, x_m, y_1, \dots, y_m\} \to V$ can be listed, one-by-one, in LOGSPACE.
- Checking if each such θ is a homomorphism can be done in NLOGSPACE.
- Checking if $\theta(\bar{z}) = \bar{t}$ is in LOGSPACE.

Unions of C2RPQs:

Suppose we want to extract pairs of authors who have a joint paper.



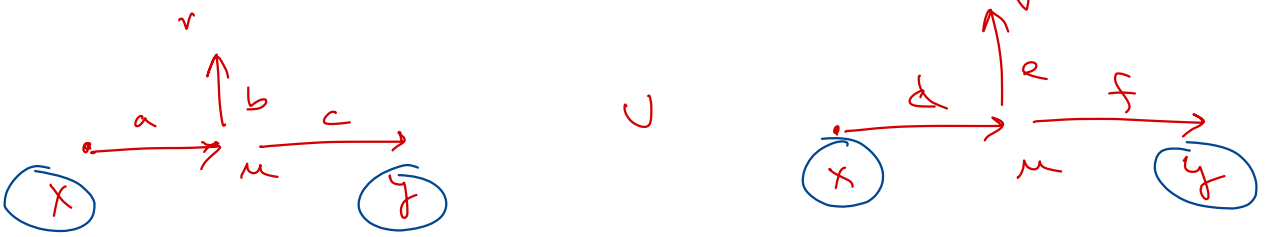
We can exploit disjunction in regular expressions to express this as a C2RPQ:

$$Ans(x,y) :- (x \xrightarrow{creator} m), (y \xrightarrow{creator} m), (m \xrightarrow{(partof \cdot series) + journal} z)$$

However, not always this "trick" can be achieved.

Example: There is no C2RPQ q that is equivalent to the union of the following C2RPQs:

1. $Ans(x,y) :- (x \xrightarrow{a} u), (u \xrightarrow{b} v), (u \xrightarrow{c} y)$
2. $Ans(x,y) :- (x \xrightarrow{d} u), (u \xrightarrow{e} v), (u \xrightarrow{f} y)$.



A union of C2RPQs is an expression $q := q_1 \cup \dots \cup q_m$, such that there exists a tuple $\bar{z}$ of variables for which

$$q_i := Ans(\bar{z}) :- \dots$$

for each $i \leq m$.

Given a graph DB $G = (V, E)$ and union of C2RPQs $q = q_1 \cup \dots \cup q_m$, we define $q(G) = \bigcup_{i=1}^{m} q_i(G)$.

EFFICIENT EVALUATION OF C2RPQs.

- Evaluation of C2RPQs taken, essentially, $|G|^{O(|q|)}$
- This is polynomial if q is fixed, but certainly not really practical if $|q| \geq 4$ and G a large graph DB.
- It'd be ideal if, for some C2RPQs, we can evaluate them in time $O(|G|^c |q|^a)$, for "small" constant c and a.

As we show next, this is the case for the class of acyclic C2RPQs. In fact, we can actually prove something stronger:

Theorem: Evaluation of acyclic C2RPQs can be solved in time $O(|G|^2 \cdot \log(|G|) \cdot |q|)$.

A C2RPQ $q := Ans(\bar{z}) :- \bigwedge_{i=1}^{m} (x_i \xrightarrow{L_i} y_i)$ is acyclic, if the undirected graph $G_q = (V = \{x_1, \ldots, x_m, y_1, \ldots, y_m\}, E = \{\{x_i, y_i\} \mid i \leq m, x_i \neq y_i\})$ is acyclic.

Example :

- $Ans(x) :- (x \xrightarrow{L_1} x), (x \xrightarrow{L_2} y), (y \xrightarrow{L_3} x)$
  is acyclic
- $Ans(x) :- (x \xrightarrow{L_1} y), (y \xrightarrow{L_2} z), (z \xrightarrow{L_3} x)$
  is not acyclic.

We now explain the procedure:

1. For each $(x_i \xrightarrow{L_i} y_i)$ in q, we compute in time $O(|G|^2 \cdot |L_i|)$ the set $Eval(x_i \xrightarrow{L_i} y_i)$ of all pairs $(u,v) \in V \times V$ s.t. there is a path ρ from u to v in G with $lab(\rho) \in L_i$.

2. For each edge $\{x_i, y_i\} \in G_q$, we compute the set:

$$Eval(\{x_i, y_i\}) = \bigcap_{(x_i \xrightarrow{L_i} y_i) \in q} Eval(x_i \xrightarrow{L_i} y_i) \cap \bigcap_{(y_i \xrightarrow{L_i} x_i) \in q} Eval(y_i \xrightarrow{L_i} x_i)^{-1}$$

Hence, $Eval(\{x_i, y_i\})$ contains all pairs $(u, v) \in V \times V$ that are linked by a path labeled in L_i, for each L_i that is required by q.

This is done in time $O(|G|^2 \cdot \log|G| \cdot |q|)$.

3. We inductively compute, using the tree-structured shape of G_q, for each variable x of q the set:

$$Hom(x) = \{ v \in V \mid \text{there is a homomorphism } \theta \text{ from } q_x \text{ to } G \text{ with } \theta(x) = v \}$$

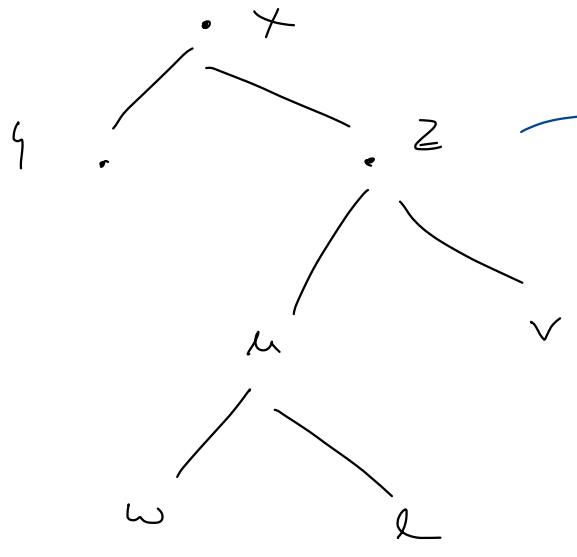
This can be done in time

$$O(|G|^2 \log |G| \cdot |q|)$$

using the following idea.

to be defined!

Join-tree of G_q:



q_z = query induced in q by the variables in the subtree rooted in z.

$$\Rightarrow q_z := Ans() :- (u \xrightarrow{L_3} z), (z \xrightarrow{L_4} v), (u \xrightarrow{L_5} w), (w \xrightarrow{L_6} v)$$

$$q := Ans() :- (x \xrightarrow{L_1} y), (z \xrightarrow{L_2} x), (u \xrightarrow{L_3} z), (z \xrightarrow{L_4} v), (u \xrightarrow{L_5} w), (w \xrightarrow{L_6} t)$$

We compute bottom-up in the join tree of q the set

$$Hom(x) = \{ v \in V \mid \text{there is a homomorphism } \theta \text{ from } q_x \text{ to } G \text{ with } \theta(x) = v \}$$

A. If x is a leaf $\Rightarrow$

$$Hom(x) = \bigcap_{x \xrightarrow{L} x \in q} Eval(x \xrightarrow{L} x).$$

This can be done in time

$$O(|G|^2 \log(|G||q|)).$$

B. If x is a node with children $x_1, \ldots, x_p$

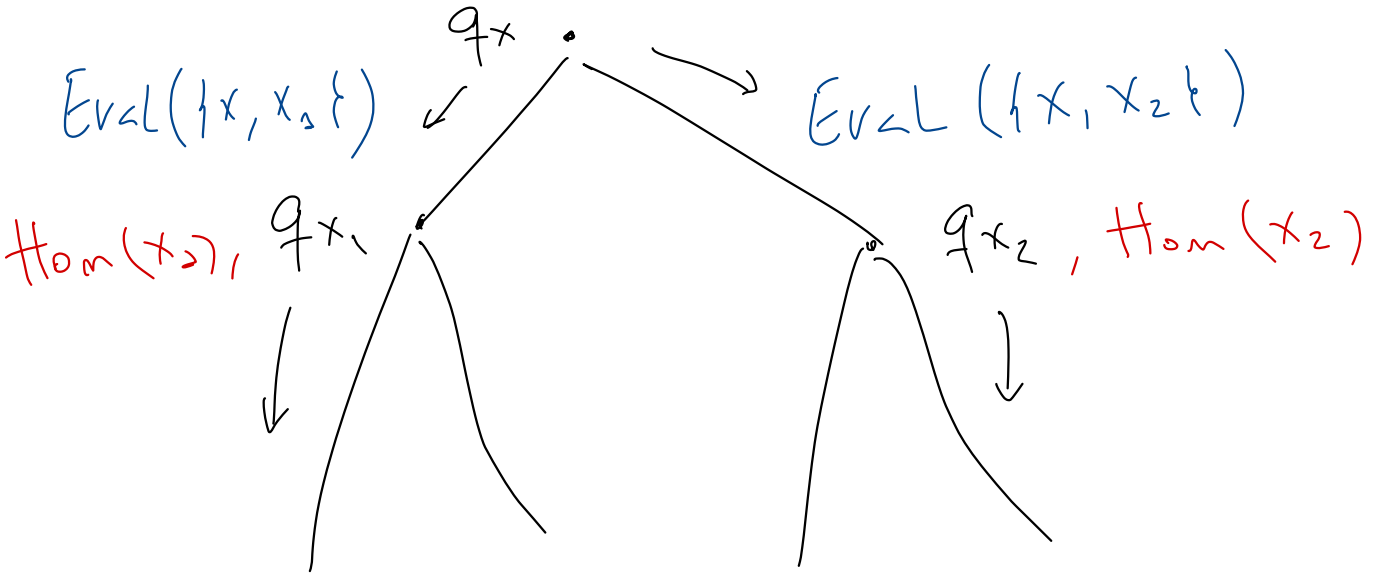
$\Rightarrow \mathrm{Hom}(x) = \widehat{\Pi}_x \alpha$, where

$$\alpha = \beta_1 \boxtimes_x \beta_2 \boxtimes_x \cdots \boxtimes_x \beta_p, \text{ and}$$

$$\beta_i = \mathrm{Eval}(\{x, x_i\}) \boxtimes_{x_i} \mathrm{Hom}(x_i).$$

This can be implemented in time $O(|G|^2 \cdot \log|G| \cdot |T|)$

Graphically :



4. We finalize checking if $\mathrm{Hom}(r) \neq \emptyset$, in which case we declare q to "hold" in G.

In fact, by definition

- $\mathrm{Hom}(r) \neq \emptyset \iff$ there is a homomorphism from qr to G.
- $qr = q$.

NESTED REGULAR EXPRESSIONS

- A C2RPQ $Ans(\bar{z}) \leftarrow \bigwedge (x_i \xrightarrow{L_i} y_i)$ is strongly acyclic, if
  1. $x_i \neq y_i$, $\forall i$ (no "loops")
  2. $\{x_i, y_i\} \neq \{x_j, y_j\}$, $\forall i,j$ with $i \neq j$. (no "parallel" edges).
- Nested regular expressions (NRE) extend the class of strongly acyclic C2RPQs with: existential tests à la XPath.
- They have been used to answer queries in RDF under RDFs schemas. [Pérez, Arenas, Gutiérrez; 2008]

A NRE over alphabet Σ is defined by the following grammar:

$$n, n' := \emptyset \mid \varepsilon \mid a \mid a^- \mid n + n' \mid n \cdot n' \mid n^* \mid \langle n \rangle$$

Example: The following are NREs over $\Sigma = \{a, b, c\}$:

$$a^* \langle b \rangle c \quad , \quad (a \langle b \rangle c)^* .$$

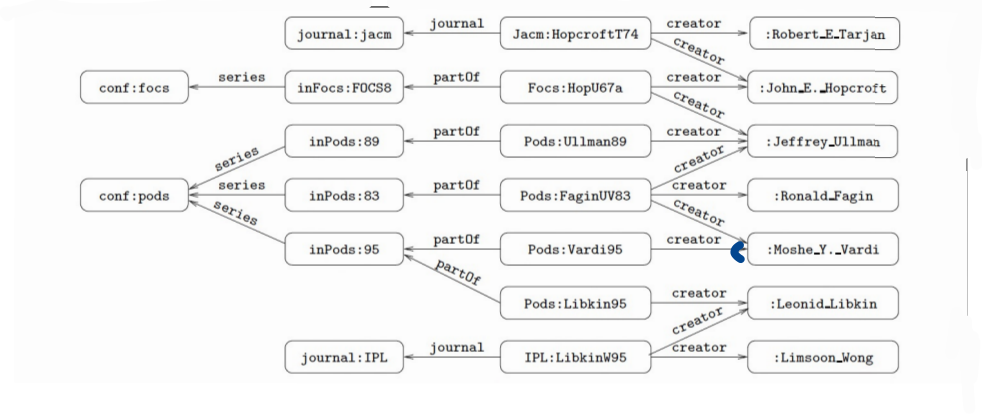
Semantics of NREs:

We define the evaluation $n(G)$ of NRE n on graph database $G = (V, E)$ inductively as follows:

- If $n = \emptyset \Rightarrow n(G) = \emptyset$
- If $n = \varepsilon \Rightarrow n(G) = id$
- If $n = a \Rightarrow n(G) = \{(u, v) \mid (u, a, v) \in E\}$
- If $n = n_1 + n_2 \Rightarrow n(G) = n_1(G) \cup n_2(G)$.
- If $n = n_1 \cdot n_2 \Rightarrow n(G) = n_1(G) \circ n_2(G)$
  $= \{(u, v) \mid \exists w : (u, w) \in n_1(G), (w, v) \in n_2(G)\}$
- If $n = n_1^* \Rightarrow n(G) = \varepsilon(G) \cup n_1(G) \cup n_1(G) \circ n_1(G) \cup \cdots$
- If $n = \langle n_1 \rangle \Rightarrow n(G) = \{(u, u) \mid \exists v : (u, v) \in n_1(G)\}$.

Example : Recall the C2RPQ :

$$Ans(x,y) :- (x \xrightarrow{creator^-} u), (y \xrightarrow{creator^-} u), (u \xrightarrow{partof \cdot series} z).$$



It can be expressed as an NRE :

$$creator^- \langle partof \cdot series \rangle creator$$

Theorem (Barceló, Pérez, Reutter; 2012)

- Every strongly acyclic C2RPQ of the form $Ans(x,y) :- \wedge(x_i \xrightarrow{L_i} y_i)$, s.t. G_q is connected, can be expressed as an equivalent NRE in which no nesting operator $\langle n \rangle$ appears as a subformula of an expression s^* (the so-called nesting-star alternation-free NREs).

Proposition: There exists a NRE over $\Sigma = \{a, b\}$ that cannot be expressed as a UC2RPQ.

Proof: Take $(a\langle b \rangle)^*$. This checks for the existence of a structure of the form:



An easy argument shows that this is not expressible as a UC2RPQ.

Theorem: The problem of checking if $(s,t) \in n(G)$, on input a graph DB $G=(V,E)$, a NRE n, and nodes s,t, can be solved in time $O(|G| \cdot |n|)$.

Proof: The nesting depth of n is the maximum number of nested existential tests $\langle p \rangle$ in n.

- If n has nesting depth 0 $\Rightarrow$ it is a regular expression and can be evaluated in time $O(|G| \cdot |n|)$.

- If n has nesting depth $k+1$
  $\Rightarrow$ it is a regular expression over the alphabet that extends Σ with NREs of the form $\langle p \rangle$, where p has nesting depth $\leq k$.
- We inductively assume that each node $v \in G$ is labeled with all such p's for which $(v, v) \in p(G)$.
- We can now evaluate n over this extension of G as if it was a RPQ.