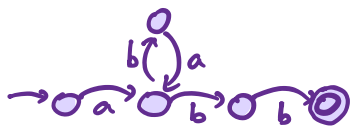


FOUNDATIONS OF GRAPH PATH QUERY LANGUAGES

Diego Figueira
CNRS, Univ. Bordeaux, LaBRI
France



I will assume some familiarity with:

- ★ Regular languages \leftrightarrow Automata 
Regular expressions $a(ba)^*bb$
- ★ Basic complexity classes

$$NL \subseteq P \subseteq NP \subseteq PSpace \neq ExpSpace$$

GRAPH DATABASES

Graph databases are used to represent **semi-structured data** in domains where the **topology** is as important as the **data** itself.

Applications in

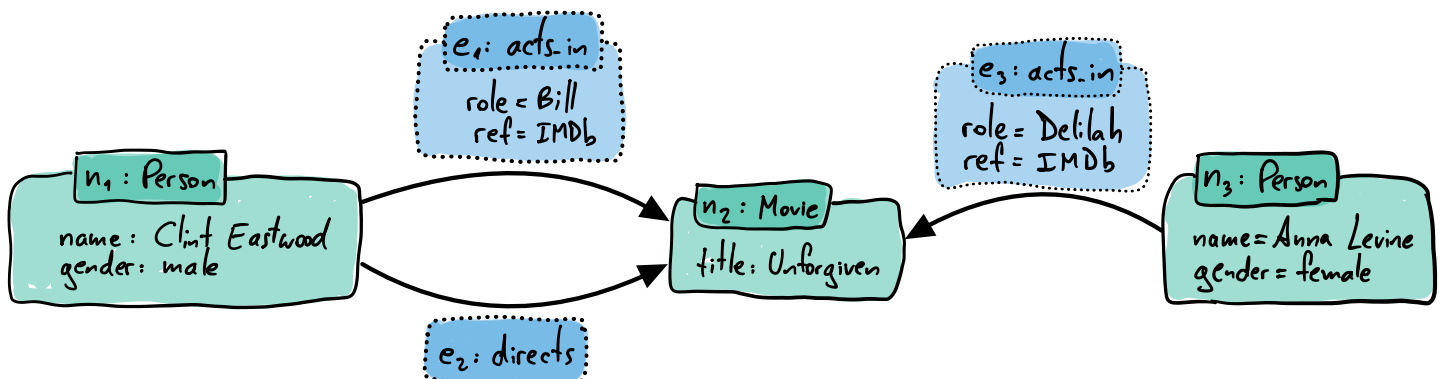
- Semantic Web
- Social Network analysis
- Biological networks
- Ontologies
- ...

DATA MODEL

Property graphs: the most general data model

- ▶ Used in commercial graph DB engines.
- ▶ Allows to represent "topology" + "properties"

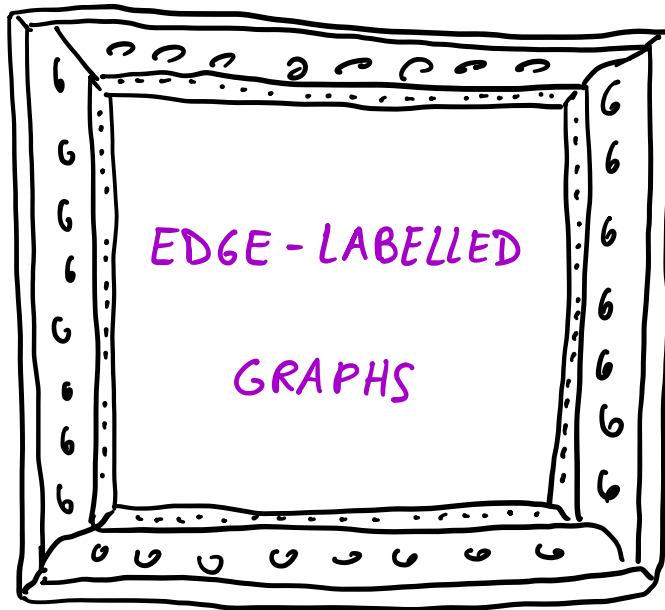
E.g.



Def. A **property graph** is a tuple $\mathcal{G} = (V, E, \varphi, \lambda, \sigma)$ where

- ▶ V is a finite set of node id's
- ▶ E is a finite set of edge id's
- ▶ $\varphi: E \rightarrow V \times V$ s.t. $\varphi(e) = (v_1, v_2)$ means " $v_1 \xrightarrow{e} v_2$ "
- ▶ $\lambda: (V \cup E) \rightarrow \text{Lab}$, where Lab is a set of "labels"
- ▶ $\sigma: (V \cup E) \times \text{Prop} \rightarrow \text{Str}$ is a partial function, where
 - Prop : set of properties
 - Str : set of strings

For theoretical exploration, we consider the simpler model of



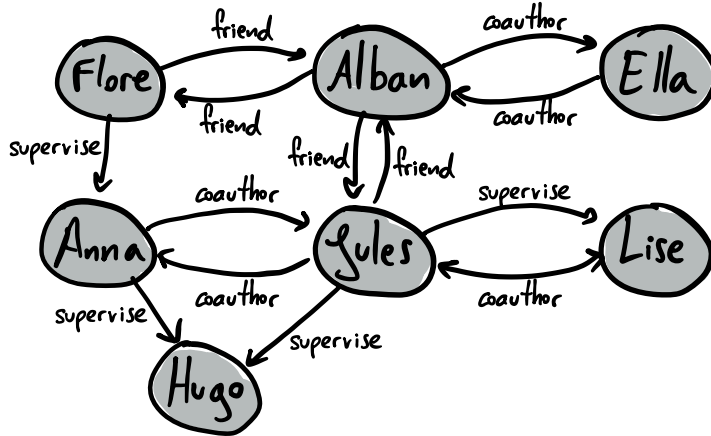
Def. A **graph database** over \mathcal{A} is a tuple $\mathcal{G} = (V, E)$ where

► V is a finite set of nodes

► $E \subseteq V \times \mathcal{A} \times V$ a set of edges labeled by \mathcal{A} .

We draw " $u \xrightarrow{a} v$ " to denote $(u, a, v) \in E$.

E.g.



$\mathcal{A} = \{ \text{friend, coauthor, supervise} \}$

REGULAR PATH QUERIES



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

RPQ: the language allowing to traverse the graph while checking whether a condition holds.

Def. A **path** in $G = (V, E)$ is a sequence

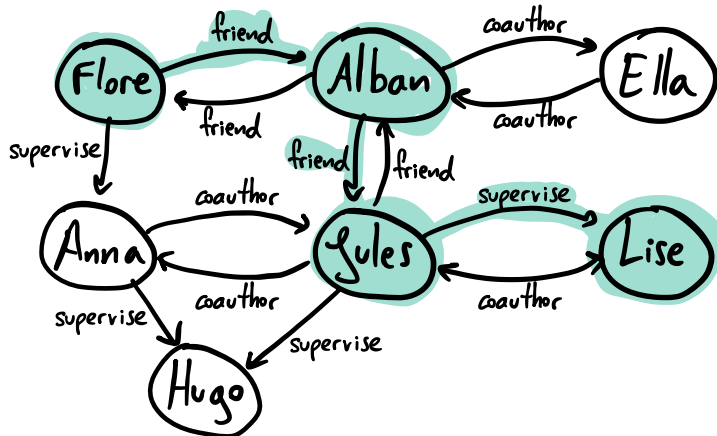
$$p = v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \xrightarrow{a_3} v_3 \xrightarrow{a_4} \dots \xrightarrow{a_{n-1}} v_{n-1} \xrightarrow{a_n} v_n$$

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

The **label** of p is $\text{lab}(p) = a_1 \dots a_n \in A^*$

($p = v$ is a path with $\text{lab}(p) = \epsilon$)

E.g.



A path with label
"friend · friend · supervise"

Def. A **regular path query (RPQ)** over \mathbb{A} is an expression

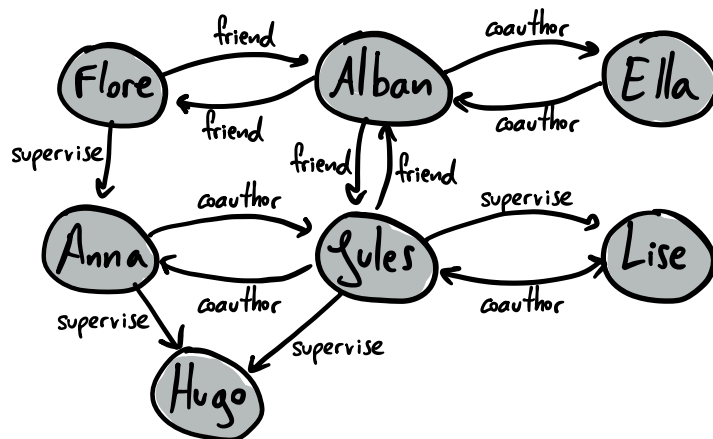
$$Q(x, y) = x \xrightarrow{L} y$$

where L is a regular expression over \mathbb{A} .

Def. Given a graph DB $\mathcal{G} = (V, E)$ the **evaluation of Q on \mathcal{G}** (denoted $Q(\mathcal{G})$) is the set

$$\{ (u, v) \in V \times V : \exists \text{ a path } p \text{ from } u \text{ to } v \text{ in } \mathcal{G} \text{ s.t. } \text{lab}(p) \in L \}$$

E.g. $Q(x, y) = x \xrightarrow{\text{coauthor}^*} y$



This is called the "arbitrary path semantics".

- Arbitrary path semantics : any path
- Simple path semantics : only simple paths, ie, no repeating nodes
- Trail semantics: paths with no repeated edges

Obs. In general: $(u, v) \in Q(Q)$ under simple path semantics



$(u, v) \in Q(Q)$ under trail semantics



$(u, v) \in Q(Q)$ under arbitrary path semantics

INVERSES

Def.

A : finite alphabet

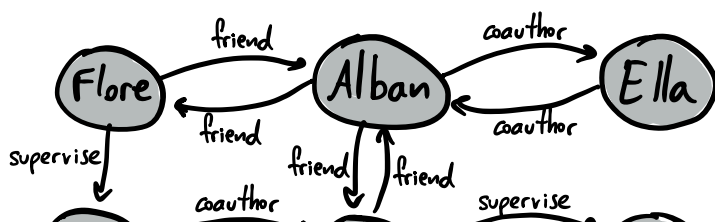
$$A^{\pm} = A \cup \{a^{-} : a \in A\}$$

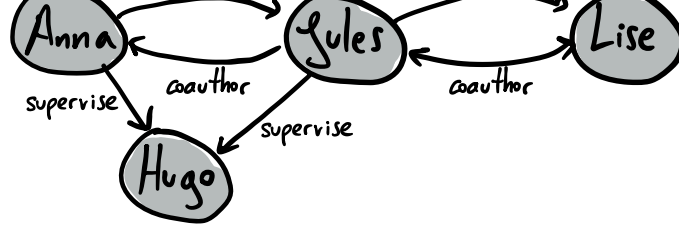
A 2RPQ over A is an expression

$$Q(x, y) = x \xrightarrow{L} y$$

where L is a regular expression over A^{\pm} .

E.g. $Q(x, y) = x \xrightarrow{\text{supervise} \cdot \text{supervise}^{-}} y$ (co-supervisor relation)





SEMANTICS

given $Q = (V, E)$ over A

let $Q^{\pm} = (V, E^{\pm})$ over A^{\pm} where

$$E^{\pm} = E \cup \{(u \xrightarrow{a} v) : (u \xrightarrow{a^{-1}} v) \in E\}$$

Def. Given a 2RPQ $Q(x, y) = x \xrightarrow{L} y$, we define $Q(Q)$ as $Q(Q^{\pm})$ (seen as an RPQ over A^{\pm})

CONJUNCTIVE RPQ

$$\frac{\text{CRPQ}}{\text{RPQ}} = \frac{\text{CQ}}{\text{Relational atoms}}$$

Use *conjunction* of RPQ and *project* onto some variables.

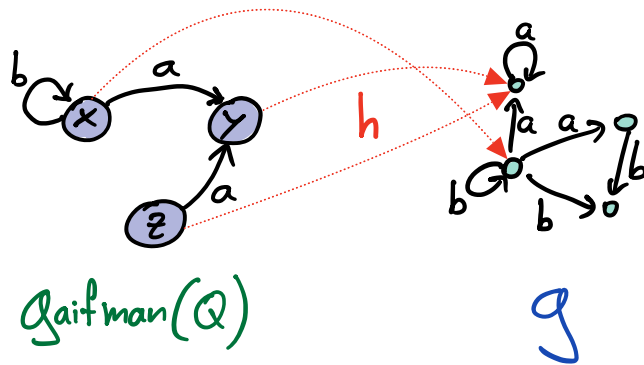
Conjunctive Query: A first-order query of the form

$$\exists z_1, \dots, z_m. R_{a_1}(x_1, y_1) \wedge \dots \wedge R_{a_n}(x_n, y_n).$$

Lemma [Chandra, Merlin '77]

$$g \models Q \text{ iff } \exists h: \text{hom } \text{gaifman}(Q) \xrightarrow{h} g$$

E.g. $Q = \exists x y z R_a(x, y) \wedge R_b(x, x) \wedge R_a(z, y)$



Now "atoms" are RPQ...

Def. A **conjunctive 2RPQ (C2RPQ)** is an expression

$$Q(z_1, \dots, z_n) = (x_1 \xrightarrow{L_1} y_1) \wedge \dots \wedge (x_m \xrightarrow{L_m} y_m)$$

where

- The x_i 's and y_i 's are variables

- z_1, \dots, z_n is a tuple of "free" variables in $\{x_1, \dots, x_m, y_1, \dots, y_m\}$

- Each L_i is a regular expression over A^\pm .

A **homomorphism** from Q to $g=(V,E)$ is a mapping μ from the variables of Q to V .

A homomorphism **satisfies** Q if $(\mu(x_i), \mu(y_i)) \in (x_i \xrightarrow{L_i} y_i)(g)$ for all i .

Finally, the evaluation $Q(g)$ is the set

$$\{(\mu(z_1), \dots, \mu(z_n)) : \mu \text{ a satisfying homomorphism from } Q \text{ to } g\}$$

E.g.

$$\triangleright Q(x) = x \xrightarrow{\text{supervise}^+} y \wedge x \xrightarrow{\text{friend}^+} y$$

"Retrieve all persons being friends with some descendant in the supervisor genealogy"

$$\triangleright Q() = x \xrightarrow{\text{supervise}^+} x$$

"There is a cycle in the supervisor genealogy"

▶▶▶ When $Q()$ is **Boolean** (= no free variables)
we write $q \models Q$ instead of $() \in Q(q)$
↑
the "empty tuple"



DECISION PROBLEMS

★ **Query evaluation**: Is $(n_1, \dots, n_t) \in Q(q)$?

★ **Static Analysis**: Does Q have a certain property?

- ★ Emptiness
- ★ Containment
- ★ Equivalence
- ★ Boundedness
- ⋮

EVALUATION OF CRPQ

Problem: eval- \mathcal{Q}

Given: $Q \in \mathcal{Q}$, a graph db \mathcal{G} , a tuple \bar{x}

Whether: $\bar{x} \in Q(\mathcal{G})$

2 flavours $\begin{cases} \text{combined complexity: both } Q, \mathcal{G} \text{ as input} \\ \text{data complexity: } Q \text{ is considered to be fixed} \end{cases}$

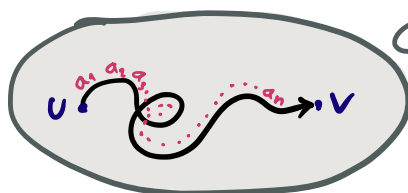
Thm.

eval-(2)RPQ is NL-complete

(both in data and combined complexity)

and in linear time $O(|Q| \cdot |L|)$.

Proof idea. Input: (u, v) , $x \xrightarrow{L} y$, $\mathcal{G} = (V, E)$

We want to check  where $a_1 \dots a_n \in L$

① turn L into an automaton A_L (in logspace / linear time)

- ② "see" Q as an automaton A_Q with initial/final states u/v
- ③ take the intersection of both (in logspace / linear time)
- ④ check for emptiness (in NL)

Thm.

eval- $C(2)RPQ$ is NP-complete in combined complexity
 NL-complete in data complexity

Proof idea for NP upper bound

- ① Guess a homomorphism $\mu: \text{vars}(Q) \rightarrow V$
 sending the free vars z_1, \dots, z_n to the input tuple.
- ② Check in polynomial time that $(\mu(x_i), \mu(y_i)) \in (x_i \xrightarrow{L} y_i)(Q)$
 (in NL)
 for every i . In other words, that μ is satisfying.

NL upper bound: same as before, noticing that since Q is fixed
 μ can be written in logspace.

NP lower bound: from eval-CQ, which is NP-complete

[Chandra, Merlin '77]

Obs. $CQ \approx CRPQ$ whose languages are of form $\{a\}$.

⇒ What about other semantics?

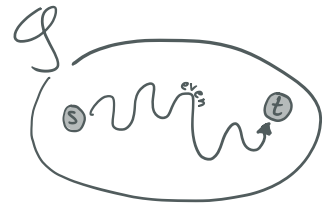
Thm. [Mendelzon, Wood '95]

eval-RPQ under simple path semantics is NP-complete, both in data and combined complexity.

Proof

Consider the RPQ $Q(x, y) = x \xrightarrow{(aa)^*} y$.

On a -labelled graphs, it checks if there is a simple path between two given vertices, which is NP-complete. [Lapough, Papadimitriou '84]



Then the query $x \xrightarrow{(aa)^*} y$ is bad (NP-c)

for simple path semantics but $x \xrightarrow{a^*} y$ is good (NL-c).

Can we tell apart bad and good queries?
And what about trail semantics?

Thm. [Bagan, Bonifati, Groz '20; Martens, Niewerth, Trautner '20]

For every RPQ $Q(x,y) = x \xrightarrow{L} y$ $\text{eval}\{Q\}$ under
simple path or trail semantics is

- ▶ NP-complete,
- ▶ NL-complete, or
- ▶ in L. (even in AC^0)

Characterizations are effective (in PSpace).



STATIC ANALYSIS OF CRPQ

bunch of reasoning tasks
for queries
in the absence of data



Static Analysis

→ of importance to optimization or for checking correctness properties.

Static Analysis Problems

- ▶ **Satisfiability**: Is there some q s.t. $q \models Q$?
- ▶ **Containment**: Is $q \models Q_1 \Rightarrow q \models Q_2$ for all q ?
- ▶ **Equivalence**: Is $q \models Q_1 \Leftrightarrow q \models Q_2$ for all q ?
- ▶ **Boundedness**: Can the expressions of the query be simplified?
Can we avoid the use of recursion?
Is there some Q' without Kleene star s.t.
 $Q \equiv Q'$?
- ▶ **Semantic Acyclicity**: Can the shape of a query be simplified?
Can cycles be avoided?
Is there some Q' without non-trivial cycles
s.t. $Q \equiv Q'$?

SATISFIABILITY

Problem: sat-Q

Given: $Q \in \mathcal{Q}$

Whether: there is some graph \mathcal{G} s.t. $Q(\mathcal{G}) \neq \emptyset$

E.g.

- sat-FO is undecidable [Trakhtenbrot '50]
- sat-PROP is NP-complete
- sat-CQ is constant time (always true!)

Obs. For any $Q \in \text{CRPQ}$ over $A = \{a_1, \dots, a_n\}$ TFAE

- (i) Q is satisfiable
- (ii) $\bigcup_{a_1, \dots, a_n} \mathcal{G}_{a_1, \dots, a_n} \models Q$
- (iii) $L \neq \emptyset$ for every lang. L in Q

Lemma

Sat-CRPA is in polynomial time. In fact:

▶ in L if languages given by regexp

▶ in NL if languages given by NFA/DFA

(Same for RPQ, UC2RPQ, ...)

- Emptiness for DFA/NFA is NL-complete
- Emptiness for regular expressions is in L



CONTAINMENT

Problem: ~~cont-Q~~

Given: $Q_1, Q_2 \in \mathcal{Q}$

Whether: $Q_1(\mathcal{G}) \subseteq Q_2(\mathcal{G}) \quad \forall \text{ graph db. } \mathcal{G}$

Why do we care about containment?

The most basic reasoning task. Useful to

Optimization Can I safely replace Q with Q' ?

Verification Can I obtain Q_{secret} from Q ?

Integrity checking Is Q between Q_{under} and Q_{over} ?

⋮

R PQ CONTAINMENT

REMEMBER

Given $Q_1 = x \xrightarrow{L_1} y$ and $Q_2 = x \xrightarrow{L_2} y$,
is $Q_1(q) \subseteq Q_2(q)$ for every graph db q ?

Lemma

$$x \xrightarrow{L_1} y \subseteq x \xrightarrow{L_2} y \text{ iff } L_1 \subseteq L_2$$

Proof

\Leftarrow If $L_1 \subseteq L_2$ and $(v_1, v_2) \in (x \xrightarrow{L_1} y)(q)$



q

$$\text{label}(p) \in L_1 \subseteq L_2 \Rightarrow (v_1, v_2) \in (x \xrightarrow{L_2} y)(q)$$

\Rightarrow) If $x \xrightarrow{L_1} y \subseteq x \xrightarrow{L_2} y$, let $w \in L_1$, for $w = a_1 \dots a_n \in A^*$.
We show $w \in L_2$.

Consider $Q := v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} v_n$. Then, $(v_0, v_n) \in (x \xrightarrow{L_1} y)(Q)$,
and thus $(v_0, v_n) \in (x \xrightarrow{L_2} y)(Q)$. Since there's only one path
from v_0 to v_n this means $a_1 \dots a_n \in L_2$.

Corollary

\subseteq -RPQ is PSpace-complete.

\rightarrow Containment pb. for
regular langs. is PSpace-c.



CRPQ CONTAINMENT

Thm. [Calvanese & al. KR'00, Florescu & al PODS'98]

The CRPQ containment problem is decidable,
ExpSpace-complete.

(Also for UC2RPQ.)

► For simplicity, let's focus on **Boolean** queries.

Given $Q_1, Q_2 \in CRPQ$, $\forall q$ ~~$Q_1(q) = Q_2(q)$~~ ?

$$\mathcal{Q} \models Q_1 \Rightarrow \mathcal{Q} \models Q_2 ?$$

▶ Visual representation:

$$Q_1 = \exists x_1 x_2 x_3 \quad x_1 \xrightarrow{(ab)^*} x_2 \wedge x_1 \xrightarrow{a^*} x_3 \wedge x_3 \xrightarrow{c^*} x_2 \wedge x_2 \xrightarrow{(a+b)^*} x_3$$

$\begin{array}{c} \text{ee} \qquad \qquad \qquad \text{,,} \\ = \qquad \begin{array}{c} x_1 \xrightarrow{(ab)^*} x_2 \\ \quad \quad \quad \uparrow \quad \downarrow \\ \quad \quad \quad c^* \quad (a+b)^* \\ \quad \quad \quad \downarrow \quad \uparrow \\ x_1 \xrightarrow{a^*} x_3 \end{array} \end{array}$

Lemma: For every $Q_1, Q_2 \in \text{CRPQ Boolean}$,

$$Q_1 \subseteq Q_2 \text{ iff } \forall Q_1^e \in \text{Exp}(Q_1) \exists Q_2^e \in \text{Exp}(Q_2) \text{ s.t. } Q_2^e \xrightarrow{\text{hom}} Q_1^e.$$

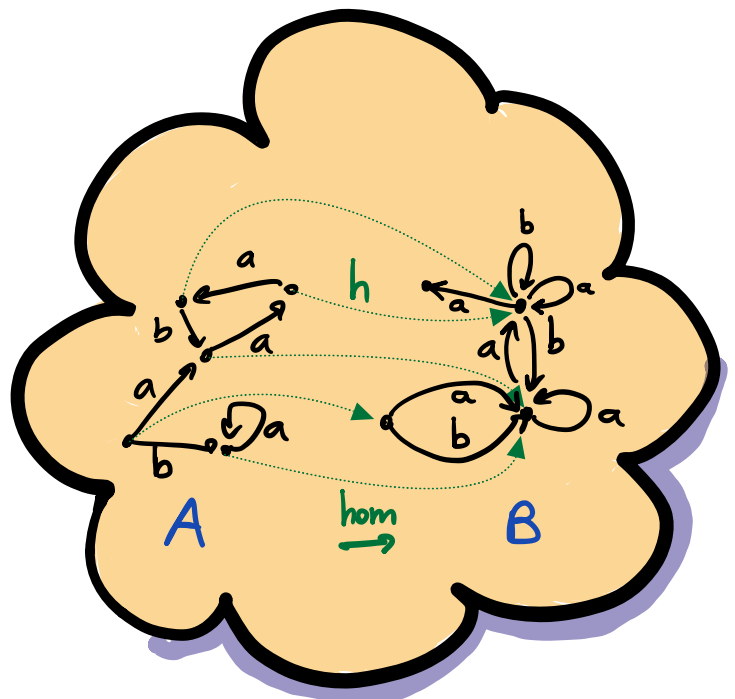
Alternatively:

$$Q_1 \not\subseteq Q_2 \text{ iff } \exists Q_1^e \in \text{Exp}(Q_1) \forall Q_2^e \in \text{Exp}(Q_2) . Q_2^e \not\xrightarrow{\text{hom}} Q_1^e.$$

Def. \hookrightarrow witness for non-containment

→ Remember:

$$A \xrightarrow{\text{hom}} B$$



Proof (Remember: for a CQ Q : $Q \models Q$ iff $Q \xrightarrow{\text{hom}} Q$)

Let $Q_1 \equiv \overbrace{(A_1 \vee A_2 \vee \dots)}^{\text{Exp}(Q_1)}$, $Q_2 \equiv \overbrace{(B_1 \vee B_2 \vee \dots)}^{\text{Exp}(Q_2)}$.

\Leftarrow) Assume $\forall i \exists j$ st $B_j \rightarrow A_i$.

$Q \models A_1 \vee \dots \Rightarrow \exists i \ Q \models A_i \Leftrightarrow A_i \rightarrow Q$

Take j st $B_j \rightarrow A_i$. Then $B_j \rightarrow A_i \rightarrow Q$, hence $Q \models B_j$.

Thus, $Q \models B_1 \vee \dots$

\Rightarrow) Assume $A_1 \vee \dots \subseteq B_1 \vee \dots$ and let's show $\forall i \exists j B_j \rightarrow A_i$.

Take any i , and let $Q = A_i$.

Since $Q \models A_1 \vee \dots$, then $Q \models B_1 \vee \dots$

$\underbrace{\text{because } Q \models A_i}_{\text{because } A_i \xrightarrow{Q} Q}$

\Downarrow
 $\exists B_j$ st $Q \models B_j$

\Downarrow
 $B_j \rightarrow Q (= A_i)$

CRPQ CONTAINMENT \leadsto UPPER BOUND

GOAL: Checking $Q_1 \subseteq Q_2$ in Exp Space



STATIC ANALYSIS

\hookrightarrow CONTAINMENT

\hookrightarrow CRPQ

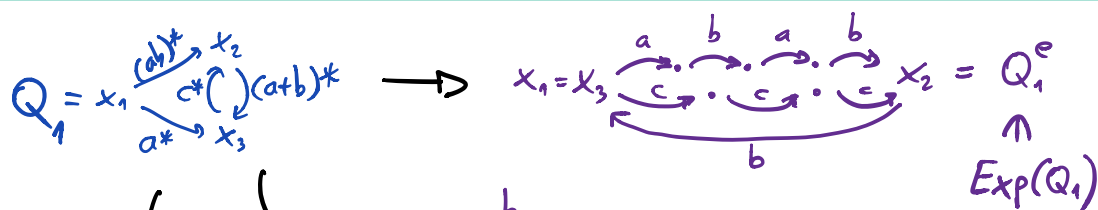
\hookrightarrow UPPER BOUND

► Consider the encoding of *expansions* into *words*

$$\text{enc} : \text{Exp}(Q_1) \rightarrow (A \cup \{\$, \} \cup \text{Vars}(Q_1))^*$$

Definition by example:

E.g.



$$\text{enc}(Q_1^e) = \$ x_1 a b a b x_2 \$ x_1 x_3 \$ x_3 c c c x_2 \$ x_2 b x_3 \$$$

Let $L_{\text{enc}} \stackrel{\text{def}}{=} \{ \text{enc}(Q^e) : Q^e \in \text{Exp}(Q_1) \} \subseteq_{\text{reg}} (A \cup \{ \$ \} \cup \text{Var}(Q_1))^*$

We test: $L_{\text{enc}} \stackrel{?}{\subseteq} \{ \text{enc}(Q^e) : Q^e \text{ is not a witness} \}$

$\underbrace{\text{poly NFA}}$
 $\underbrace{\text{exp-sized NFA}}$

$\underbrace{\hspace{15em}}_{\text{exp space}}$

\approx "no expansion of Q_1 is a witness for non-containment"

\approx " $\forall E_1 \in \text{Exp}(Q_1) \exists E_2 \in \text{Exp}(Q_2) \text{ s.t. } E_2 \rightarrow E_1$ "

for each $\text{enc}(Q_i^e)$, we need to test the existence of a homomorphism

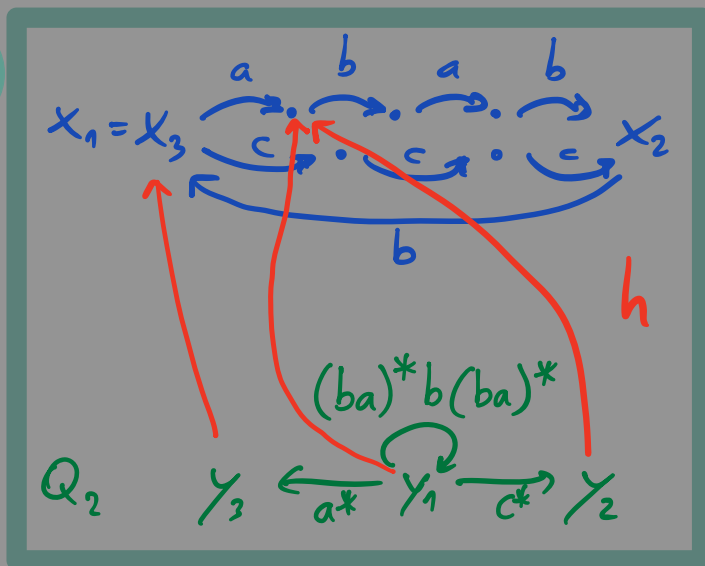
$$\text{Exp}(Q_2) \ni Q_2^e \rightarrow Q_1^e$$

In $Q_2^e \in \text{Exp}(Q_2)$ has 2 sorts of nodes $\begin{cases} \text{nodes from } \text{Var}(Q_2) \\ \text{other nodes.} \end{cases}$

We encode assignments

$$h: \text{Var}(Q_2) \rightarrow Q_1^e$$

E.g.



Given $Q_1^e \in \text{Exp}(Q_1)$, $h: \text{Vars}(Q_2) \rightarrow \text{Vars}(Q_1^e)$

We define $\text{enc}(Q_1^e, h) \in (A \cup \{\$, \} \cup \text{Vars}(Q_1) \cup 2^{\text{Vars}(Q_2)})^*$

defn by example:

E.g. $\text{enc}(Q_1^e, h) = \$ x_1 a \{y_1, y_2\} b a b x_2 \$ x_1 x_3 \$ x_3 \{y_3\} c c c x_2 \$ x_2 b x_3 \$$

$$\uparrow$$

$$(A \cup \{\$, \} \cup \text{Vars}(Q_1) \cup \cancel{2^{\text{Vars}(Q_2)}})^*$$

$\{\$, \} \cup \text{Vars}(Q_2)$
(to avoid exp. alphabet)

We encode h by inserting the preimage of each node

$$\text{map } L_{\text{enc}} := \{\text{enc}(Q^e, h) : Q^e \in \text{Exp}(Q_1), h: \text{Vars}(Q_2) \rightarrow \text{Vars}(Q^e)\} \subseteq_{\text{reg}} (\dots)^*$$

(Basically: $\text{enc}(Q_1^e) + \text{scattered partition of } \text{Vars}(Q_2)$)



source of an exponential

It can be described by a poly 2NFA

two way NFA

2NFA

$$\delta = \{(q, a, \rightarrow, q'), (q', b, \rightarrow, q'), (q', b, \leftarrow, q''), \dots\}$$

$\vdash a b a a b b a b \vdash$

SUB-GOAL: Build an automaton for...

$$L_{Q_2} = \{ \text{enc}(Q_1^e, h) \in L_{\text{enc}}^{\text{map}} : Q_2^e \xrightarrow{g} Q_1^e \text{ for some } Q_2^e \in \text{Exp}(Q_2) \text{ s.t. } h \in g \}$$

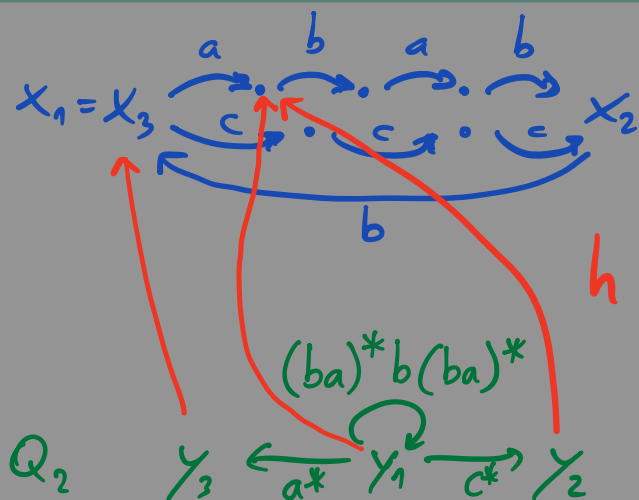
HOW? Given an atom $y \xrightarrow[\text{(edge)}]{S} y'$ of Q_2 ,

$$L_{y \xrightarrow{S} y'} := \{ \text{enc}(Q_1^e, h) \in L_{\text{enc}}^{\text{map}} : Q_1^e, (h(y), h(y')) \models y \xrightarrow{S} y' \}$$

$L_{\text{poly 2NFA}}$

"there's an S-path $h(y) \rightsquigarrow h(y')$ in Q_1^e "


E.g.



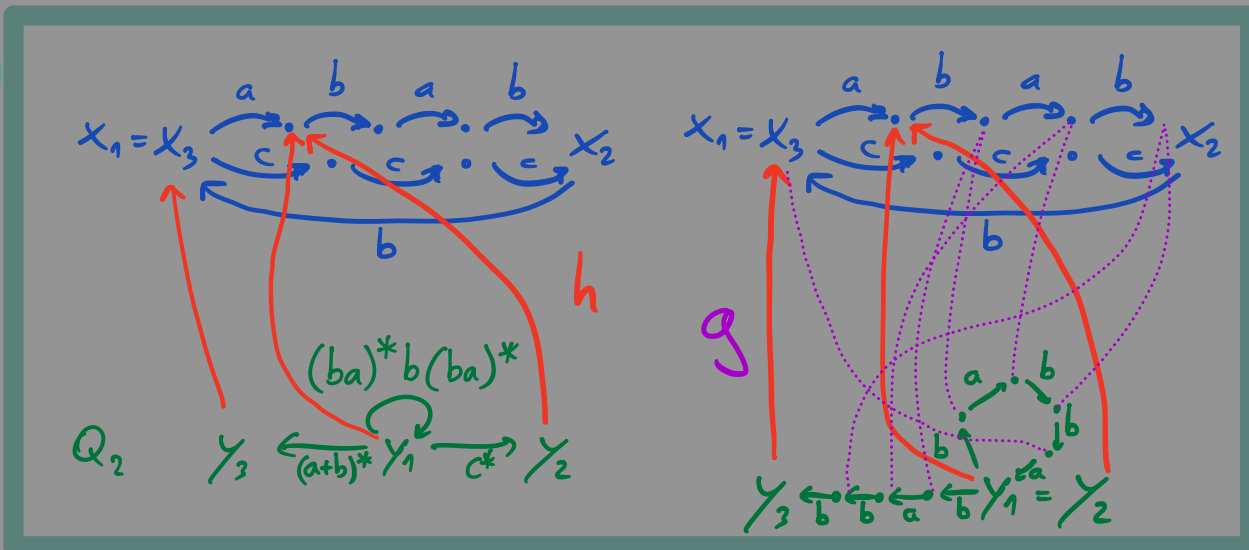
$$\text{enc}(Q_1^e, h) = \$ x_1 a \{x_1, x_2\} bab x_2 \$ x_1 x_3 \$ x_3 \{y_3\} ccc x_2 \$ x_2 b x_3 \$$$

$$\text{atom } y_1 \xrightarrow{(ba)^* b (ba)^*} y_1 \text{ of } Q_2$$

Define: $L_{Q_2} := \bigcap_{y \in Q_2} L_y$ (poly 2NFA)

Obs. $L_{Q_1} = \{ \text{enc}(Q_1^e, h) \in L_{\text{enc}}^{\text{map}} : Q_2^e \xrightarrow{g} Q_1^e \text{ for some } Q_2^e \in \text{Exp}(Q_2) \text{ s.t. } h \subseteq g \}$ sub-GOAL 

E.g.



Now define

$$L_{Q_2}^{\pi} := \prod_{A \cup \text{Vars}(Q_1) \cup \{ \$ \}} (L_{Q_2}) \quad (\text{exp. NFA})$$

Obs. $L_{Q_2}^{\pi} = \{ \text{enc}(Q_1^e) : Q_1^e \in \text{Exp}(Q_1) \text{ is } \underline{\text{not}} \text{ a witness for non-containment} \}$

$$Q_1 \subseteq Q_2 \iff L_{\text{enc}} \subseteq L_{Q_2}^{\pi}$$

poly exp
ExpSpace

▶ **free vars. allowed:** we consider only mappings which are **identity** on free vars. (assuming $\text{free}(Q_1) = \text{free}(Q_2)$).

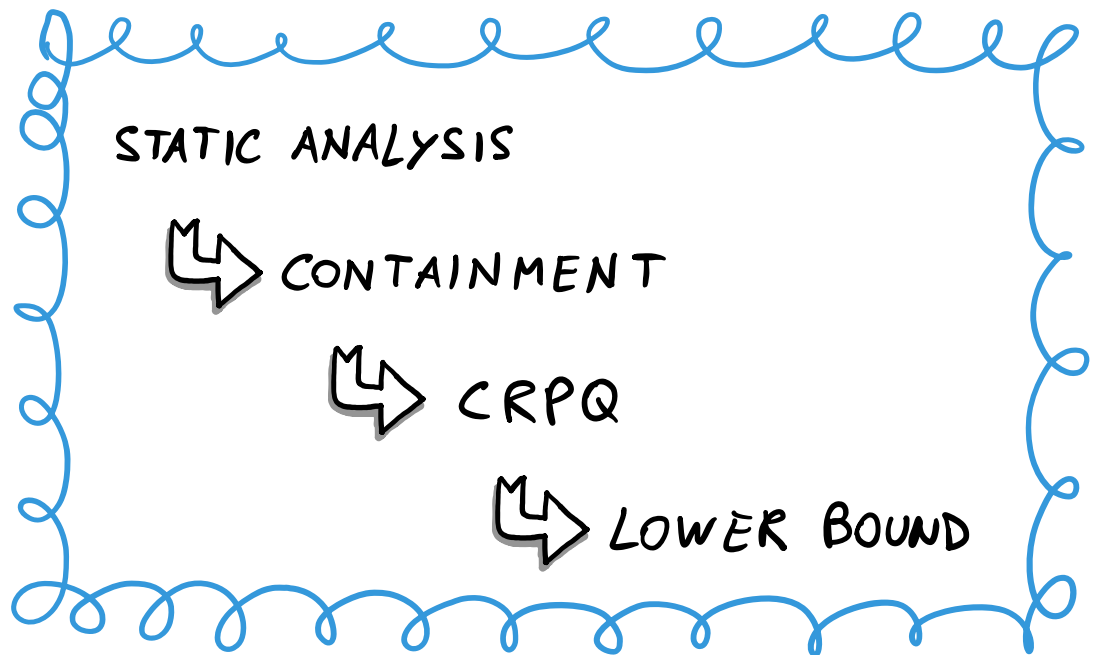
▶ **two-wayness (CZRPQ):** 2NFA can follow paths in 2 directions.

▶ **Unions (UCRPQ):** $\text{Exp}' = \bigcup_{i \in I} \text{Exp}_i$



CRPQ CONTAINMENT AND LOWER BOUND

GOAL: \subseteq -CRPQ is ExpSpace-hard



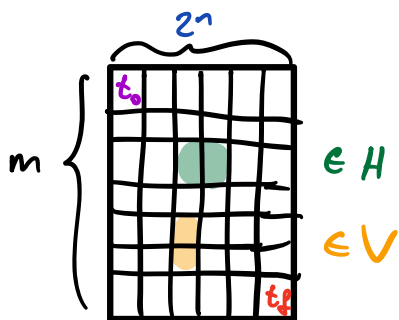
Reduction from ExpSpace-complete

"Exponential Corridor Tiling problem"

Def. Exponential Corridor Tiling problem:

- Input:
- $n \in \mathbb{N}$ (in unary)
 - T : finite set of "tiles"
 - $H, V \subseteq T \times T$
 - $t_0, t_f \in T$

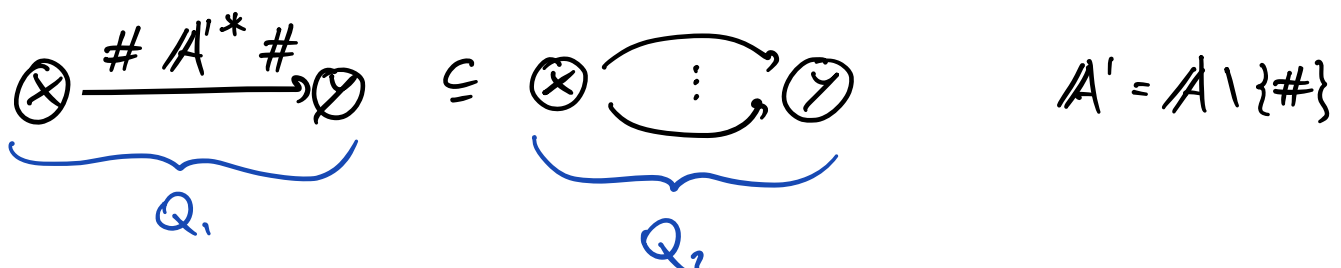
► Question: $\exists m \exists m \times 2^n$ tiling $\varphi: \{1, \dots, m\} \times \{1, \dots, 2^n\} \rightarrow T$
 verifying constraints H, V ?



→ Encoding of a tiling solution (eg $n=4$)

#0000 $t_{1,1}$ 0001 $t_{1,2}$ 0010 $t_{1,3}$... 1111 $t_{1,2^4}$ \$
 0000 $t_{2,1}$ $t_{2,2^4}$ \$
 : : :
 : : :
 $t_{m-1,2^4}$ \$
 $t_{m,2^4}$ # $\in (T \cup \{0,1,\$, \#\})^*$
 \mathbb{A}

$Q_1 \subseteq Q_2 \iff$ no successful tiling
 ↳ boolean ↳ \iff no $w \in \mathbb{A}^*$ is the encoding of a successful tiling



$L_{\text{enc}} := \{ w \in A^* : \#w\# \text{ not the encoding of a tiling} \}$
 \nwarrow incrementing error
 \nearrow miss placed \$
 \vdots bit string too long/short

$$L_{\text{init}} := \bigcup_{t \neq t_0} \# 0^n t$$

$$L_{\text{fin}} := \bigcup_{t \neq t_f} t \#$$

$$L_H := \bigcup_{(t, t') \notin H} t (0+1)^n t'$$

$$\begin{aligned}
 L_V &:= \bigcup_{(t, t') \notin V} \bigcap_{1 \leq i \leq n} \bigcup_{j \in \{0,1\}} (0+1)^{i-1} j (0+1)^{n-i} t (A' \setminus \{\$ \})^* \$ \\
 &\quad (A' \setminus \{\$ \})^* (0+1)^{i-1} j (0+1)^{n-i} t' \\
 &= \bigcap_{1 \leq i \leq n} \bigcup_{(t, t') \notin V} \bigcup_{j \in \{0,1\}} (0+1)^{i-1} j (0+1)^{n-i} t (A' \setminus \{\$ \})^* \$ \\
 &\quad (A' \setminus \{\$ \})^* (0+1)^{i-1} j (0+1)^{n-i} t' \\
 &\quad \underbrace{\hspace{15em}}_{L_{V,i}}
 \end{aligned}$$

$$Q_2 := \bigotimes \left(\begin{array}{c} L_{V,1} \cup L_H \cup L_{\text{init}} \cup L_{\text{fin}} \cup L_{\text{enc}} \\ \vdots \\ L_{V,n} \cup L_H \cup L_{\text{init}} \cup L_{\text{fin}} \cup L_{\text{enc}} \end{array} \right)$$

► If $w \in \#A'\#$ is not an encoding of successful tiling $\Rightarrow \exists Q_2^e \in \text{Exp}(Q_2)$ st $Q_2^e \xrightarrow{\text{hom}} x \xrightarrow{w} y$.

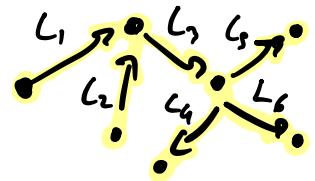
► If $w \in \#A'\#$ is a successful encoding but there was an $Q_2^e \in \text{Exp}(Q_2)$ that embeds into $x \xrightarrow{w} y$, it would witness an "error". ⚡



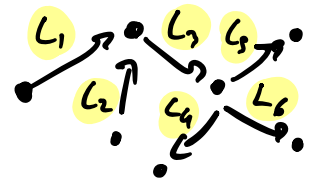
CRPQ CONTAINMENT ~ SUBCLASSES

Two natural ways to define subclasses of CRPQ:

① Restrict the "shape" of queries

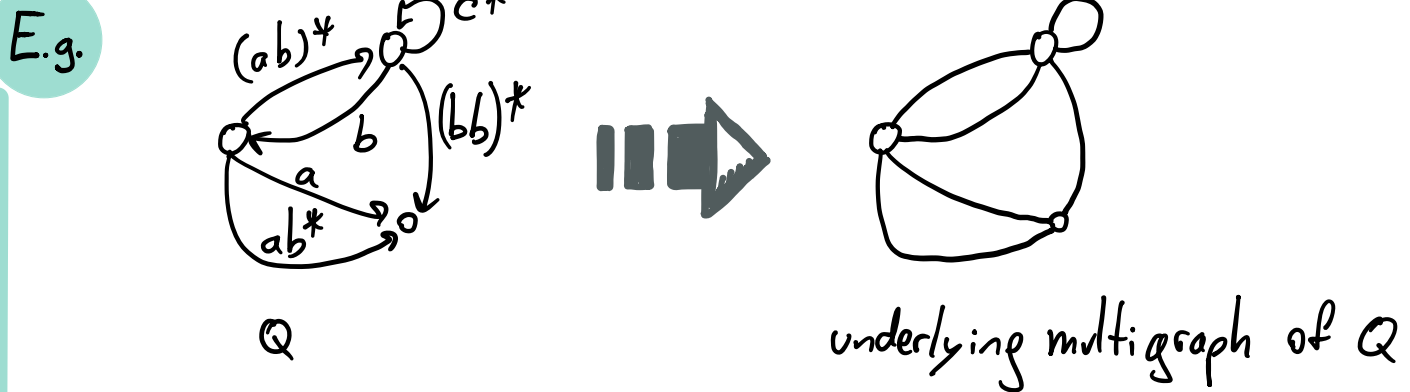


② Restrict the regular expressions



① Restrict the "shape" of queries

Def. The **underlying multigraph** of a CRPQ
=
The one obtained by **disregarding** regexp's.

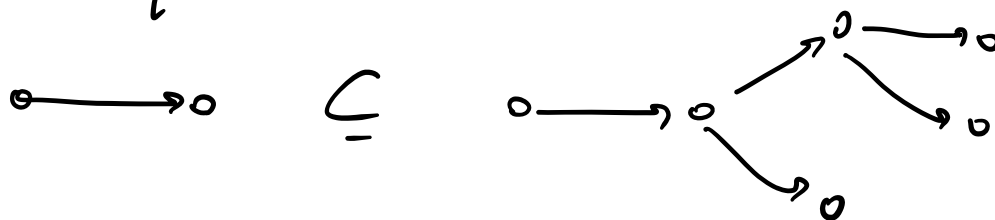


Def. For a class \mathcal{Z} of multigraphs,

$$\text{CRPQ}(\mathcal{Z}) := \{Q \in \text{CRPQ} : \text{underlying multigraph of } Q \text{ is in } \mathcal{Z}\}$$

▶ tree-like queries

E.g.



We don't need to guess the variable assignment in L_{enc}^{map} in the alphabet, we rather guess "on the fly" using **alternation**.

⇒ We obtain some $\underbrace{L_{Q_2}^\pi}_{poly\ 2AFA} \subseteq \underbrace{L_{enc}}_{poly\ NFA}$

$L_{Q_1}^\pi = \{enc(Q_1^e, h) : Q_1^e \in Exp(Q_1), h: Vars(Q_1) \rightarrow Vars(Q_1^e)\}$

$L_{Q_1}^\pi = \{enc(Q_1^e) : Q_1^e \in Exp(Q_1) \text{ is not a witness for non-containment}\}$

We need to answer:
 $L_{enc} \cap L_{Q_2}^\pi \stackrel{?}{=} \emptyset$
 "there's a Q_1 -expansion which is a witness for non-containment"
 (iff $Q_1 \not\subseteq Q_2$)

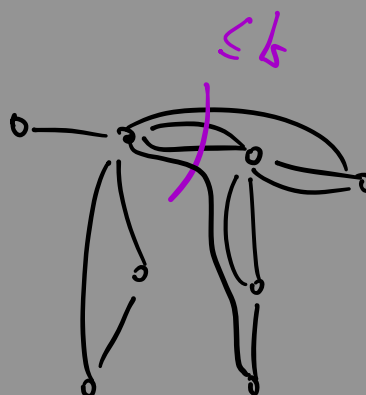
⚡ complement

$\underbrace{L_{Q_2}^\pi}_{exp\ NFA} \cap \underbrace{L_{enc}}_{pol\ NFA} \stackrel{?}{=} \emptyset$
 (under a large brace)
 PSpace (on the fly)

▶ Idem for



or



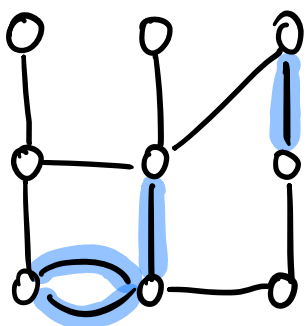
► ...or any class of graphs as far as no matter where every "bridge" (= minimal cut) is of bounded size.

Def. A **bridge** in a multigraph is a minimal set of edges whose removal increases the number of connected components.

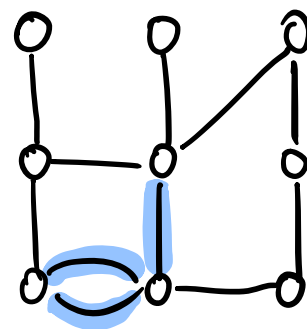
The **bridgewidth** of \mathcal{G} (noted $bw(\mathcal{G})$) = max. size of bridge in \mathcal{G}

The **bridgewidth** of a class \mathcal{C} : $bw(\mathcal{C}) = \sup_{\mathcal{G} \in \mathcal{C}} bw(\mathcal{G})$

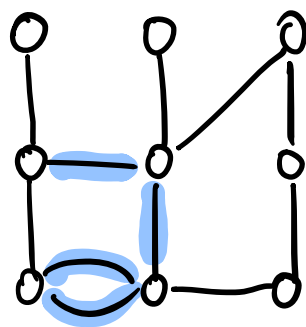
E.g.



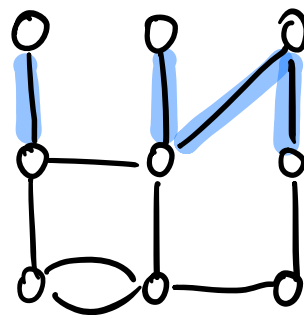
A bridge of size 4



Not a bridge
(not a separator)



Not a bridge
(not minimal)



Not a bridge
(not minimal)

Obs. $bw(G) \geq k \Leftrightarrow G$ contains $\overset{k}{\bigcirc} \circ$ as minor.

Hence: $\subseteq\text{-CRPQ}(\mathcal{C})$ is PSpace if $bw(\mathcal{C}) \leq k$.

On the other hand, if $\forall k$ we can find $\overset{k}{\bigcirc} \circ$ as a minor of some $G \in \mathcal{C}$, then we can reproduce the ExpSpace lower bound.

This yields:

Thm. [Figueira '20]

For any class \mathcal{C} of multi-graphs under mild assumptions, the $\text{CRPQ}(\mathcal{C})$ containment problem is

▶ PSpace-complete if $bw(\mathcal{C}) < \infty$, or

▶ ExpSpace-complete otherwise.

② Restrict the regular expressions

→ Very often CRPQs are used with simple regular expressions.

Like $\star a$ for $a \in \mathcal{A}$ T_a

$\star a_1 + \dots + a_n$ for $a_1, \dots, a_n \in \mathcal{A}$ T_A

$\star a^*$ for $a \in \mathcal{A}$ T_{a^*}

$\star (a_1 + \dots + a_n)^*$ for $a_1, \dots, a_n \in \mathcal{A}$ T_{A^*}



One can then define subclasses of CRPQ based on the allowed types of expressions.

E.g.

► $\text{CRPQ}(T_a) \equiv \text{CQ}$

► every $Q \in \text{CRPQ}(T_A)$ is equivalent to a UCQ

► $\text{CRPQ}(T_a) \subseteq \text{CRPQ}(T_A \cup T_{a^*})$

\mathcal{F}	$\mathcal{F} \subseteq \mathcal{F}$	$\mathcal{F} \subseteq \text{CRPQ}$	$\text{CRPQ} \subseteq \mathcal{F}$
T_a	NP (\dagger)	NP (4.2)	Π_2^p (4.4)
T_A	Π_2^p (4.3)	Π_2^p	PSpace (4.5)
$T_a \cup T_a^*$	Π_2^p (\ddagger)	Π_2^p	PSpace (5.3)
$T_A \cup T_A^*$	Π_2^p	Π_2^p (5.2)	PSpace (5.5)
$T_a \cup T_A^*$	EXPSpace (6.1)	EXPSpace	EXPSpace
$T_A \cup T_A^*$	EXPSpace	EXPSpace (\star)	EXPSpace (\star)

Table 2: Complexity of Containment of different fragments \mathcal{F} of CRPQs. All results are complete for the class given. We provide references in round brackets. When there is no bracket, the result follows directly from another cell in the table. (\dagger): (Chandra and Merlin 1977), (\ddagger): (Deutsch and Tannen 2002, fragment (l^*)), (\star): (Calvanese et al. 2000)



BOUNDEDNESS

Problem: **bound- \mathbb{Q}**

Given: $Q \in \mathbb{Q}$

Whether: $Q \equiv Q'$ for some $Q' \in UCQ$

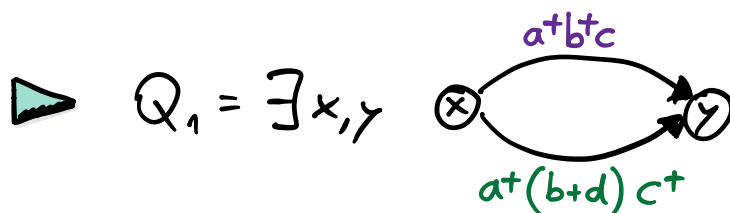
Another way to see the problem:

Remember: $Q \equiv \bigcup_{E \in \text{Exp}(Q)} E$.

→ Is there a **finite subset** $E \subseteq_{\text{fin}} \text{Exp}(Q)$ st

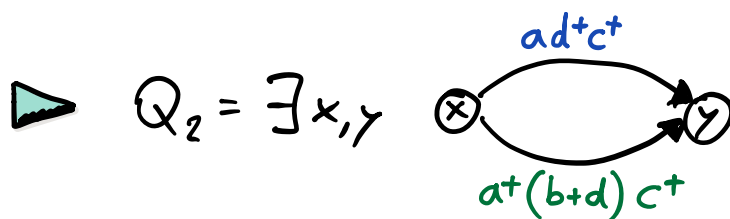
$$Q \equiv \bigcup_{E \in E} E ?$$

E.g.



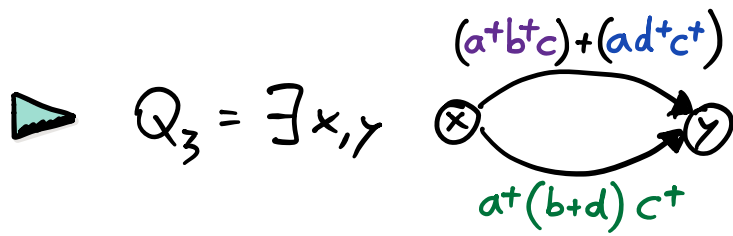
BOUNDED?

NO



NO

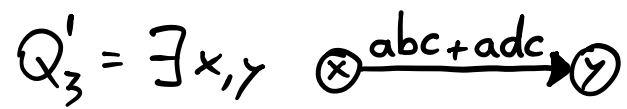




YES



equivalent to $\cap \cup$



Let's start with an easy one: Atomic queries.

R PQ BOUNDEDNESS

Q: When is $q(x, y) := x \xrightarrow{L} y$ bounded?

A: it is bounded iff L is finite!

Q: When is $q(x) := \exists y \ x \xrightarrow{L} y$ bounded?

A:

Why? i) $\exists y \ x \xrightarrow{L} y \equiv \bigcup_{u \in L} \exists y \ x \xrightarrow{u} y$

ii) $\exists y \ x \xrightarrow{u} y \subseteq \exists y \ x \xrightarrow{v} y$ iff $v \preceq_{\text{pref}} u$

i) + ii) : $\exists y \ x \xrightarrow{L} y \equiv \bigcup_{u \in L_{\text{pref}}} \exists y \ x \xrightarrow{u} y$

\uparrow pairwise incomparable

Lemma

- $x \xrightarrow{L} y$ is bounded iff $|L| < \infty$
- $\exists x \ x \xrightarrow{L} y$ is bounded iff $|L^{\text{prefix}}| < \infty$
- $\exists y \ x \xrightarrow{L} y$ is bounded iff $|L^{\text{suffix}}| < \infty$
- $\exists x, y \ x \xrightarrow{L} y$ is bounded iff $|L^{\text{factor}}| < \infty$

where

$$L^{\leq} := \{\omega \in L : \nexists \omega' \in L \text{ s.t. } \omega' \neq \omega \text{ and } \omega' \leq \omega\}$$

Lemma

Checking if L^{pref} is finite is PSpace-complete

Proof...idea for the upper bound. (same ideas for L^{suffix} , L^{factor})

From a NFA \mathcal{A} of L and a NFA \mathcal{B} of

$$L' = \{u.v : u \in L, v \neq \epsilon\}$$

(words with proper prefix in L)

Consider NFA \mathcal{B}' for

$$L^{\text{pref}} = L \cap (L')^c \quad (\text{of exponential size})$$

Check if \mathcal{B}' is finite "on the fly", ie in NL wrt $|\mathcal{B}'|$

"
 $NPSpace$ wrt $|\mathcal{A}|$

"(Savitch)

$PSpace$ wrt $|\mathcal{A}|$

Then:

Thm. [Barceló, Figueira, Romero '19]

- Boundedness for RPQs is **NL-complete**

- Boundedness for CRPQs of the form

- i) $\exists y \quad x \xrightarrow{L} y,$
- ii) $\exists x \quad x \xrightarrow{L} y,$ or
- iii) $\exists x, y \quad x \xrightarrow{L} y$

is **PSpace-complete**.

What about the general case?

Thm. [Barceló, Figueira, Romero '19]

The Boundedness problem for CRPQ is **ExpSpace-complete**.

(also for UC2RPQ)



Reduction to limitedness for distance automata.

DISTANCE AUTOMATA

- ▷ NFA with 2 sorts of transitions $\begin{cases} \text{costly} \\ \text{non-costly} \end{cases}$
- ▷ cost of run = sum of costly transitions therein
- ▷ cost of word = min. cost of accepting run

\mathcal{A} is limited if $\exists n \in \mathbb{N}$ every word in lang. of \mathcal{A} has cost $< n$.

Limitedness problem: Is a given dist. aut. limited?

↳ PSpace-complete [Hashiguchi '82; Leung, Poddsky '91 '04]

SEMANTIC ACYCLICITY

REMEMBER

$\text{eval-CRPQ}(\mathcal{C})$ is in $PTime$ iff $tw(\mathcal{C}) < \infty$
(assuming $FPT \neq W[1]$)

with self-loops and // edges

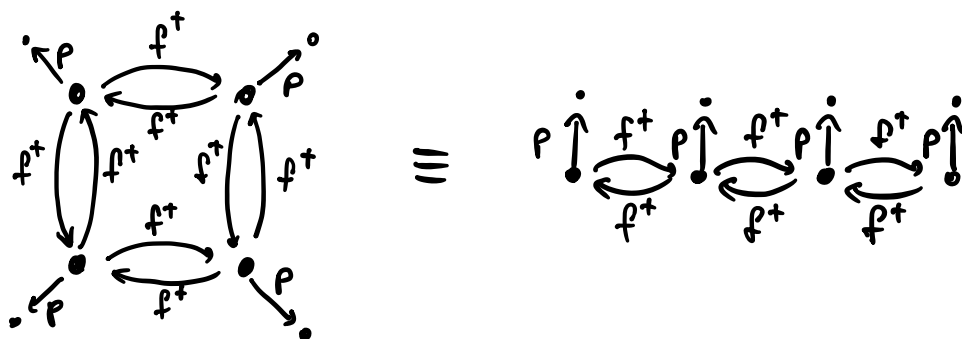
In particular: $\text{eval-CRPQ}(\text{Trees})$ is in $PTime$.

Acyclic CRPQ = a CRPQ whose underlying simple graph has no nontrivial cycles.

E.g. $Q =$  Q is Acyclic.

→ Can a query $Q \in CRPQ$ that does not look like a tree be rewritten into an acyclic one?
Yes, it can happen!

E.g.



Semantic Acyclicity problem

Problem: **SemAcyclicity - \mathcal{Q}**

Given: $Q \in \mathcal{Q}$

Whether: $Q \equiv Q'$ for some acyclic $Q' \in \mathcal{Q}$

Thm. [Barceló et al '16]

SemAcyclicity-CRPQ is ExpSpace-complete.
(Also for UC2RPQ.)



Given a UCRPQ Q , produce a UCRPQ Q^{app} , where

$Q^{app} = \underbrace{\text{the maximal acyclic underapproximation of } Q}_{\substack{\downarrow \\ \text{unique}}} \underbrace{\hspace{10em}}_{\substack{\downarrow \\ Q^{app} \subseteq Q}}$

i.e.

- 1) Q^{app} is an acyclic UCRPQ
- 2) $Q^{app} \subseteq Q$
- 3) for every acyclic $Q' \in \text{UCRPQ}$ s.t. $Q' \subseteq Q$, we have $Q' \subseteq Q^{app}$.

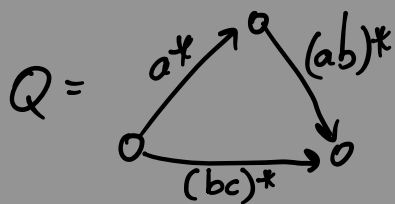
Now, a CRPQ Q is Semantically Acyclic if $Q \equiv Q^{app}$.

Why does such Q^{app} exist? And how to produce it?

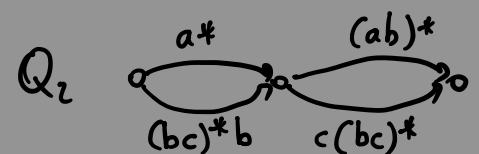
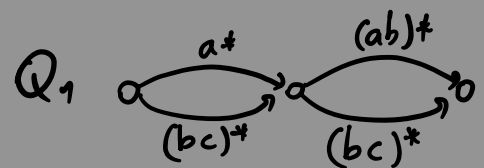


List all the "types" in which the query can be arranged in the shape of a tree.

E.g.



only bandedly many



⋮

Define $Q^{app} = \bigcup_i Q_i$.



RECAP:

- R_{PQ} & C_{RPQ} : fundamental query languages
for querying topology + labels of graph db's.
- C_{RPQ} \approx CQ + basic recursion
 \approx pattern with navigation
- evaluation **ok**, but static analysis generally **hard**
- Complexity depends on the notion of "path" we take

Nice surveys on graph query languages by :

▶ Barceló PODS'13

▶ Wood SIGMOD Rec. '12

▶ Angles, Gutierrez ACM Comput. Surv. '08

▶ Angles, Arenas, Barceló, Hogan, Reutter, Vrgoč
ACM Comput. Surv. '17

THANK YOU