Foundational aspects of

# Graph Data Management

Wim Martens
University of Bayreuth

EPIT Spring School on Theoretical Computer Science

Luminy, 2019

# Outline

- Graph Data Model
- Queries
- Graph Query Evaluation
- Graph Query Containment

- Graphs vs Trees
- "Real Queries"
- Data Value Comparisons

# Notation

# Notation and Basic Principles

If $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$

## Finite Automata

We denote a **nondeterministic finite automaton (NFA)** as

$$N = (S, A, \delta, I, F)$$

where

- $S$ is the finite set of states
- $A$ is the finite alphabet
- $\delta \subseteq S \times A \times S$ is the transition relation
- $I \subseteq S$ is the set of initial states
- $F \subseteq S$ is the set of accepting states

The **language of $N$** is denoted $L(N)$

# Notation and Basic Principles

## Regular Expressions

Operators:

(1) Kleene star (denoted *)

(2) concatenation (omitted in notation)

(3) disjunction (denoted +)

Priorities of operators: first (1), then (2), then (3)

Example: $ab+cd^*$

The **language of regular expression *r*** is denoted $L(r)$

We use $r^n$ to abbreviate $n$-fold concatenation of $r$

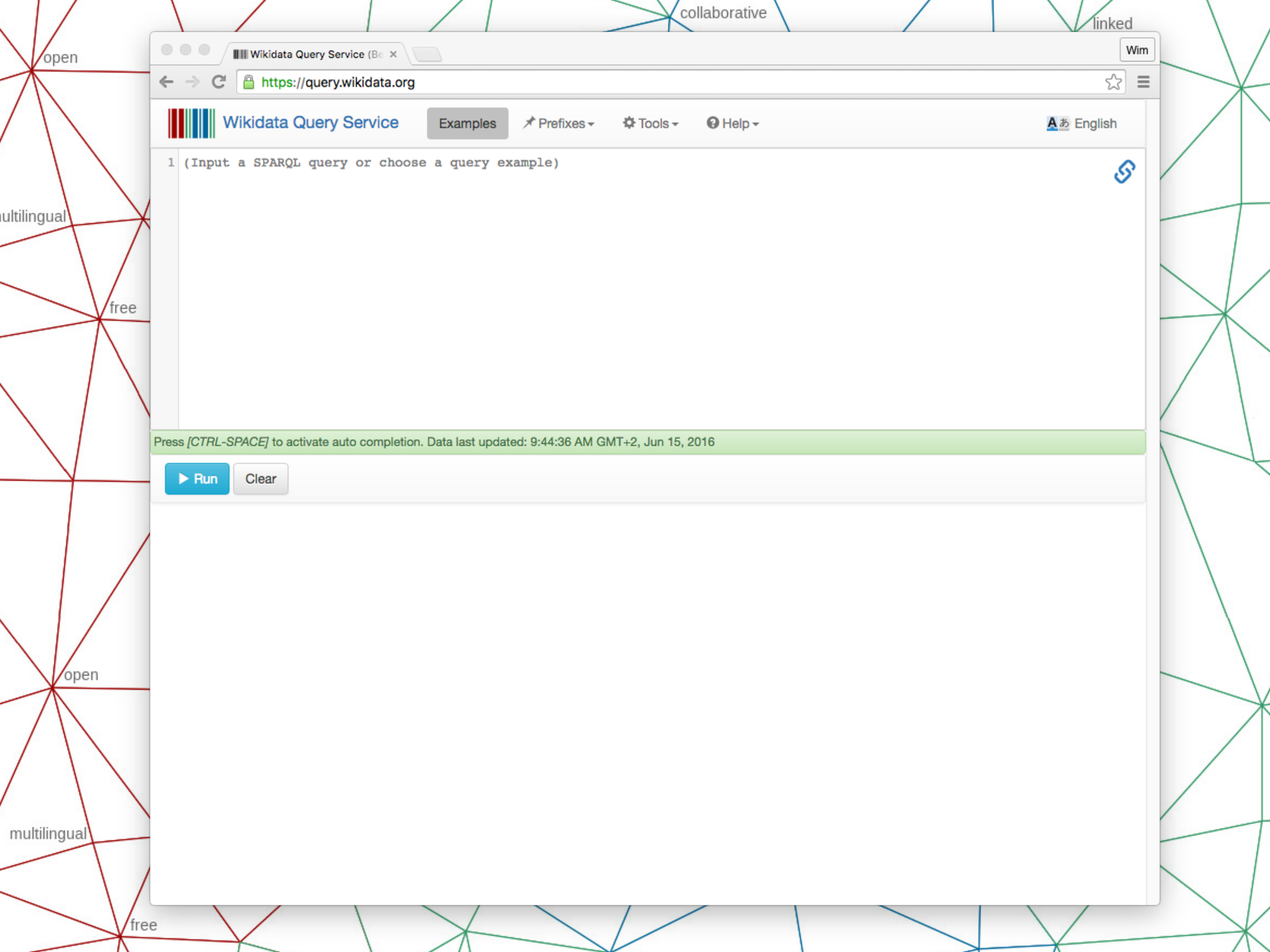
# Motivation

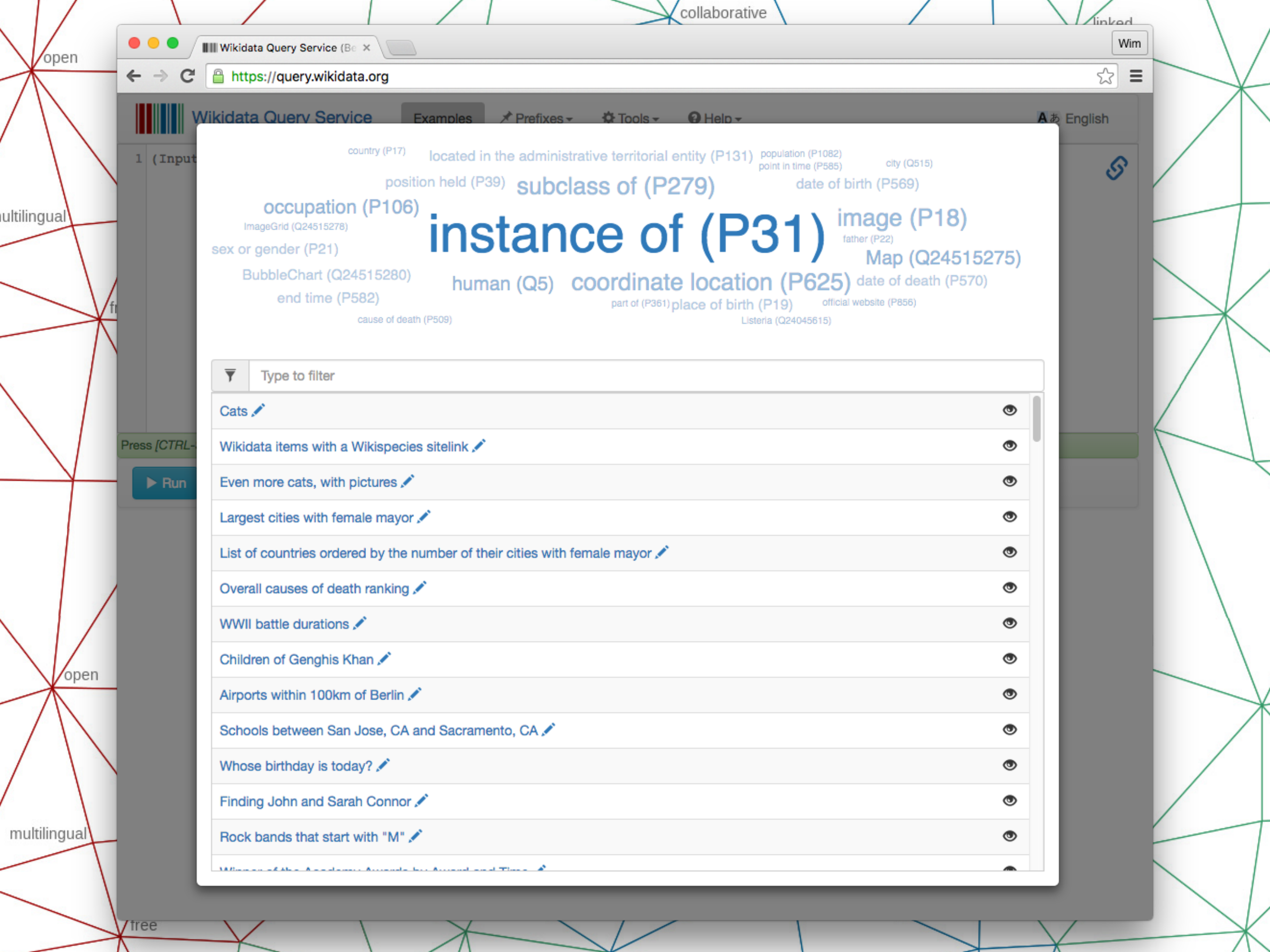
# Why Graph Databases?

- Graph databases are becoming more and more standard in industry
  [Neo4j, Tigergraph, Oracle, ...]
- They bring "reasoning about connectedness" to the masses (*)

...and they are a nice source of theory problems

(*) I heard this pitch from Hassan Chafi, Oracle


collaborative
linked
open
ultilingual
free
Wikidata Query Service (Be
Wim
https://query.wikidata.org
Wikidata Query Service
Examples
Prefixes
Tools
Help
English
1 (Input a SPARQL query or choose a query example)
Press [CTRL-SPACE] to activate auto completion. Data last updated: 9:44:36 AM GMT+2, Jun 15, 2016
Run
Clear
open
multilingual
free


collaborative
open
Wikidata Query Service (Be
Wim
https://query.wikidata.org
Wikidata Query Service
Examples
Prefixes
Tools
Help
English
1 (Input
country (P17)
located in the administrative territorial entity (P131)
population (P1082)
point in time (P585)
city (Q515)
position held (P39)
subclass of (P279)
date of birth (P569)
occupation (P106)
ImageGrid (Q24515278)
instance of (P31)
image (P18)
father (P22)
sex or gender (P21)
Map (Q24515275)
BubbleChart (Q24515280)
human (Q5)
coordinate location (P625)
date of death (P570)
end time (P582)
part of (P361)
place of birth (P19)
official website (P856)
cause of death (P509)
Listeria (Q24045615)
Type to filter
Cats
Wikidata items with a Wikispecies sitelink
Even more cats, with pictures
Largest cities with female mayor
List of countries ordered by the number of their cities with female mayor
Overall causes of death ranking
WWII battle durations
Children of Genghis Khan
Airports within 100km of Berlin
Schools between San Jose, CA and Sacramento, CA
Whose birthday is today?
Finding John and Sarah Connor
Rock bands that start with "M"
Press [CTRL-
Run
open
multilingual
free

collaborative

linked

open

ultilingual

free

structured

# Wikidata: "US artists who died of poisoning"

```
SELECT ?x
WHERE
{
    ?x wdt:occupation/wdt:subclassof* wd:artist .
    ?x wdt:citizenship wd:United_States .
    ?x wdt:cause_of_death ?y .
    ?y wdt:subclass_of* wd:poisoning
}
```

open

multilingual

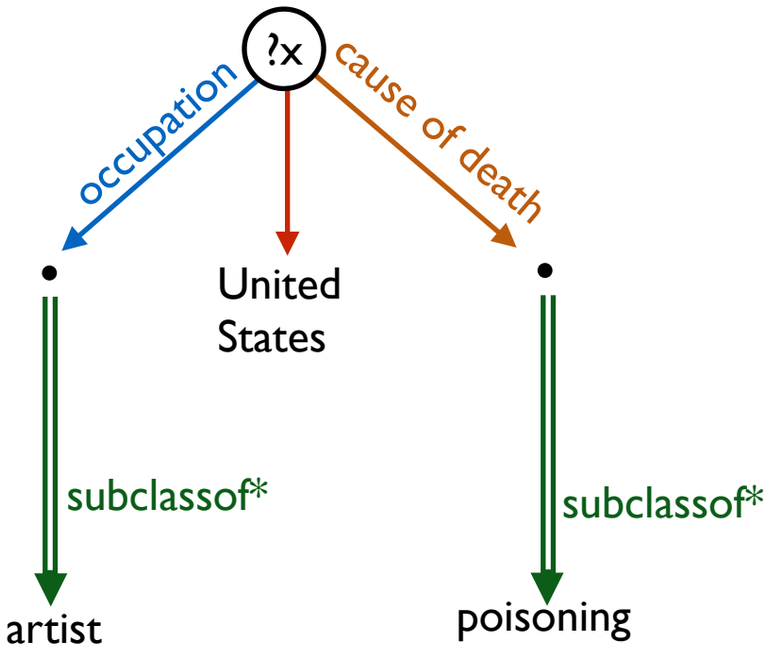
free

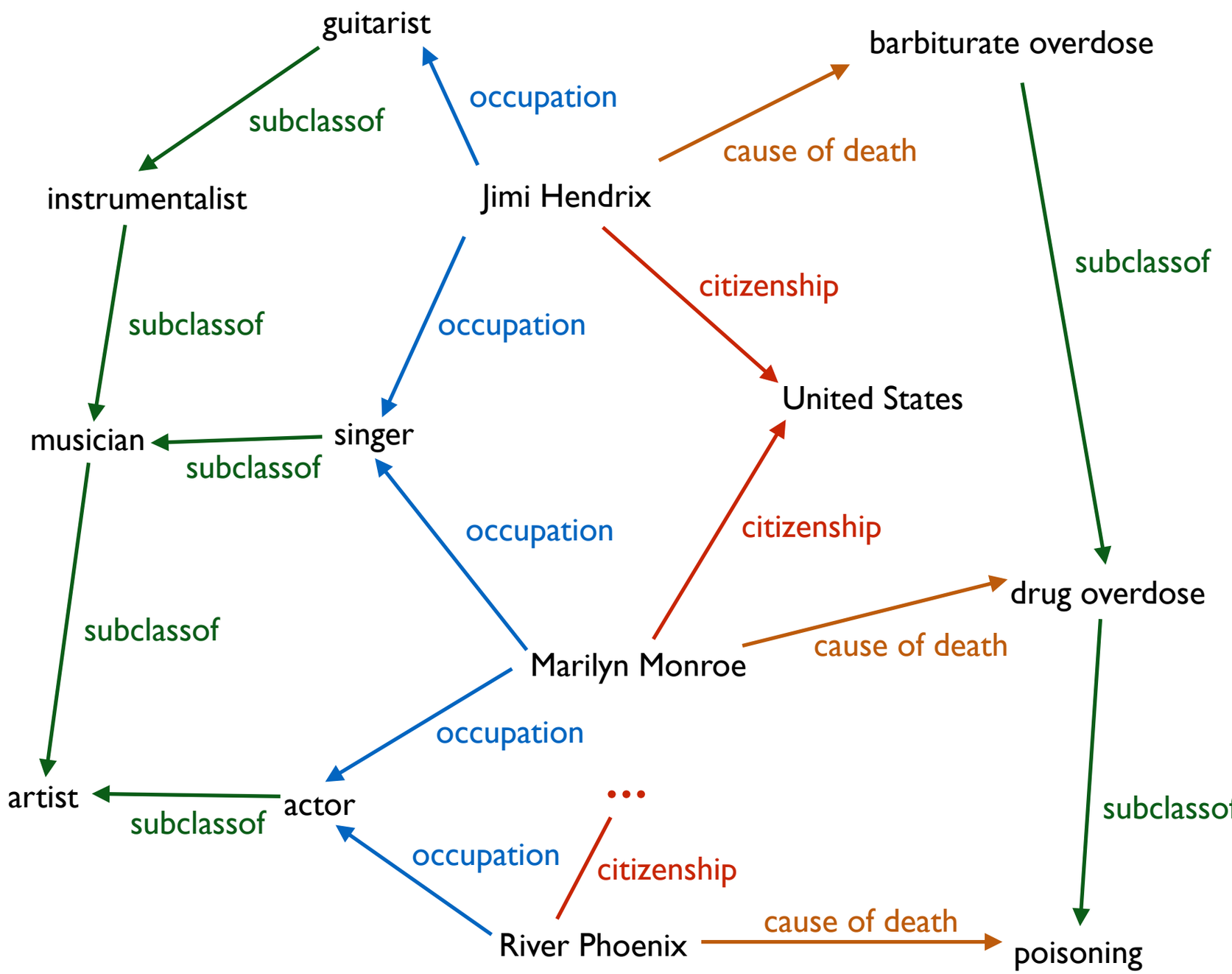
structured

(*): Original Wikidata query: politicians who died of cancer
https://www.mediawiki.org/wiki/Wikibase/Indexing/SPARQL_Query_Examples#Politicians_who_died_of_cancer_.28of_any_type.29

# Graph Queries By Example

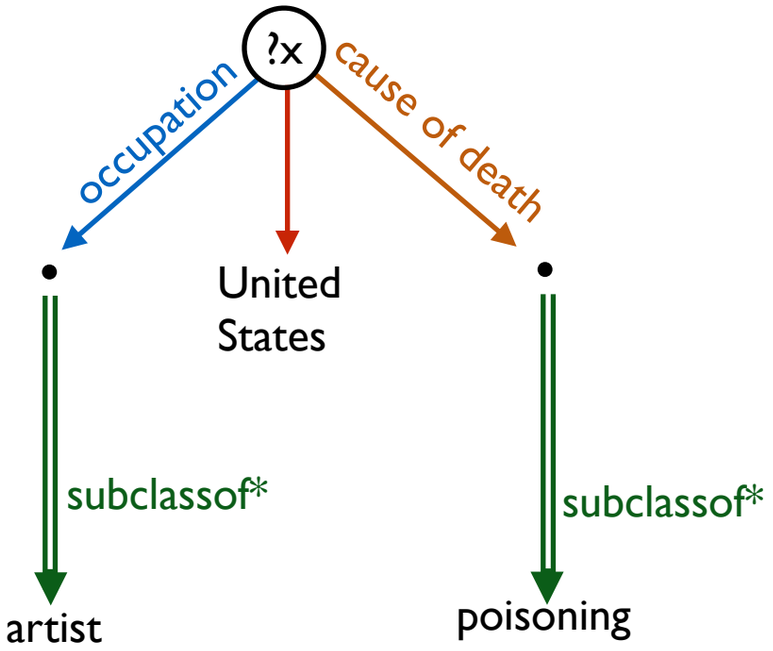
## Wikidata: "US artists who died of poisoning"

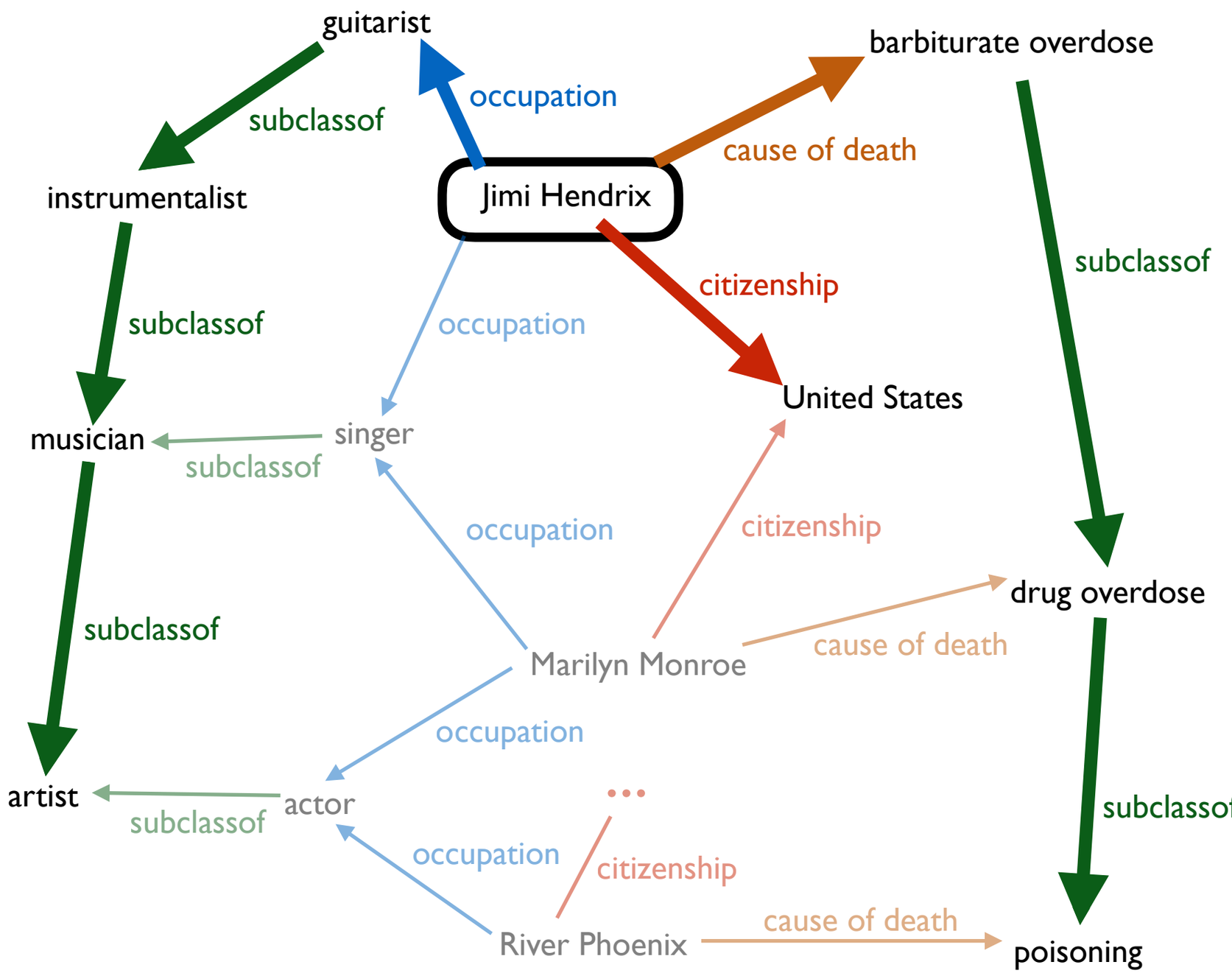


# Graph Queries By Example

## Wikidata: "US artists who died of poisoning"



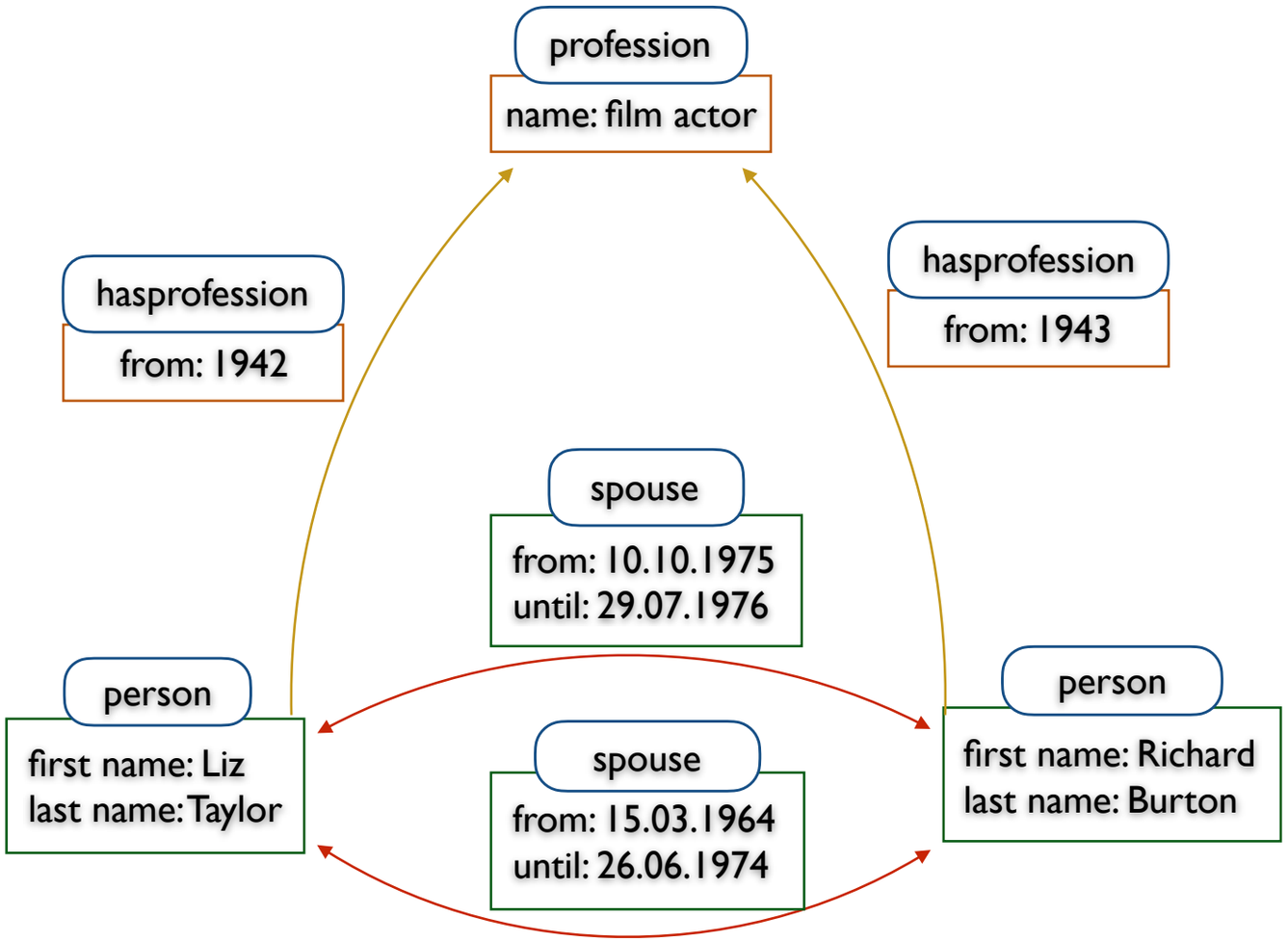
# Data Model

# What are Graph Databases?

Currently, two main data models:

- Property Graph-like Databases
- RDF-like Databases

# Property Graph Data Model

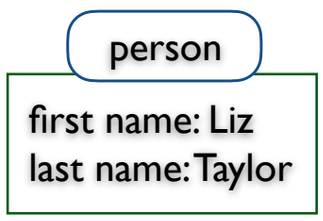


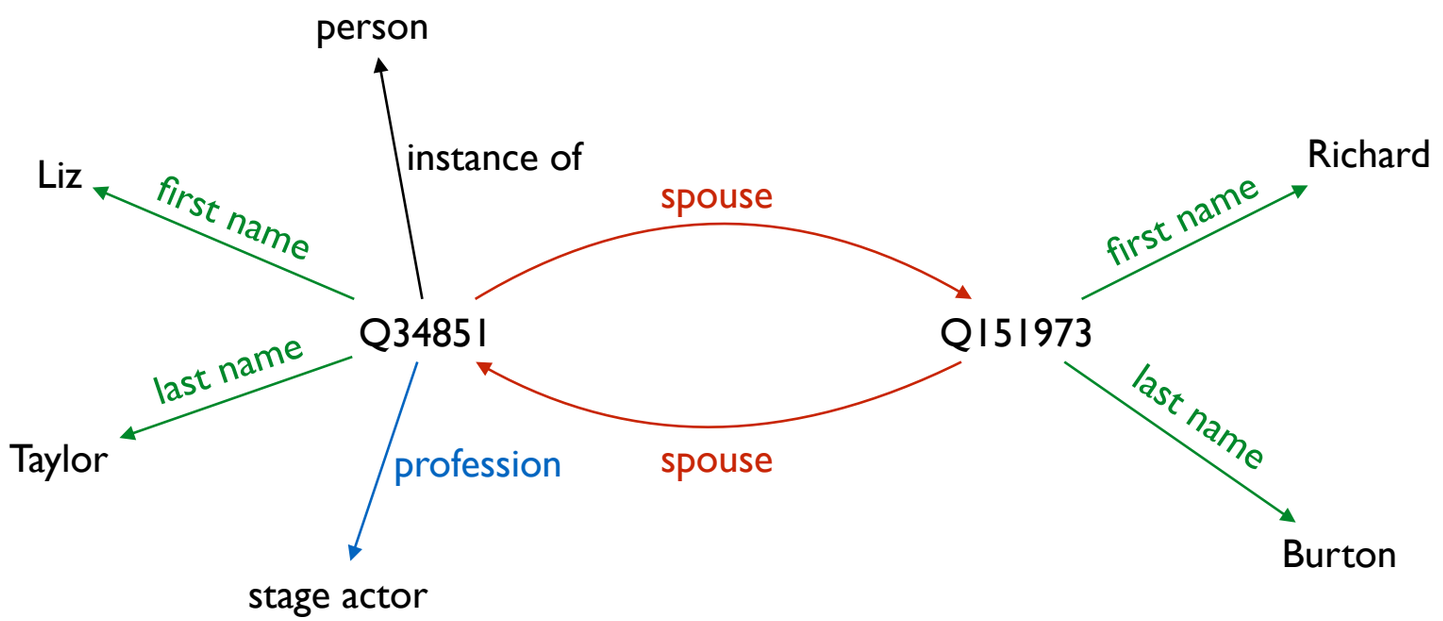

Labels **L**: person, profession, spouse
Values **V**: Liz, Taylor, 10.10.1975
Properties **P**: first name, last name

More formally, this is
- a set of node identifiers N
- a set of edge identifiers E
- a function that maps E to $N \times N$
- a function from $N \cup E$ to (subsets of) labels **L**
- a function from $(N \cup E) \times \mathbf{P}$ to (subsets of) values **V**

The G-Core model also directly incorporates a third set, containing paths [Angles et al., SIGMOD'18]

# RDF Data Model




More formally, this is a set of triples from

$$I \times I \times (I \cup L)$$

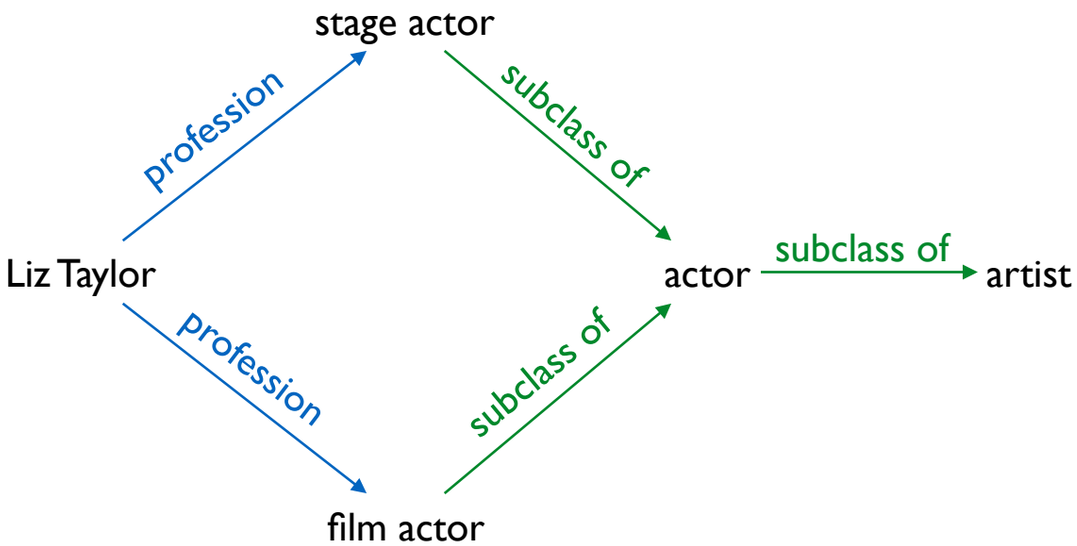
where

- $I$ is the set of Internationalized Resource Identifiers (IRIs)
- $L$ is the set of literals (constants)

These triples (s,p,o) are referred to as subject / predicate / object triples

(There are also *blank nodes*)

# RDF Data Model

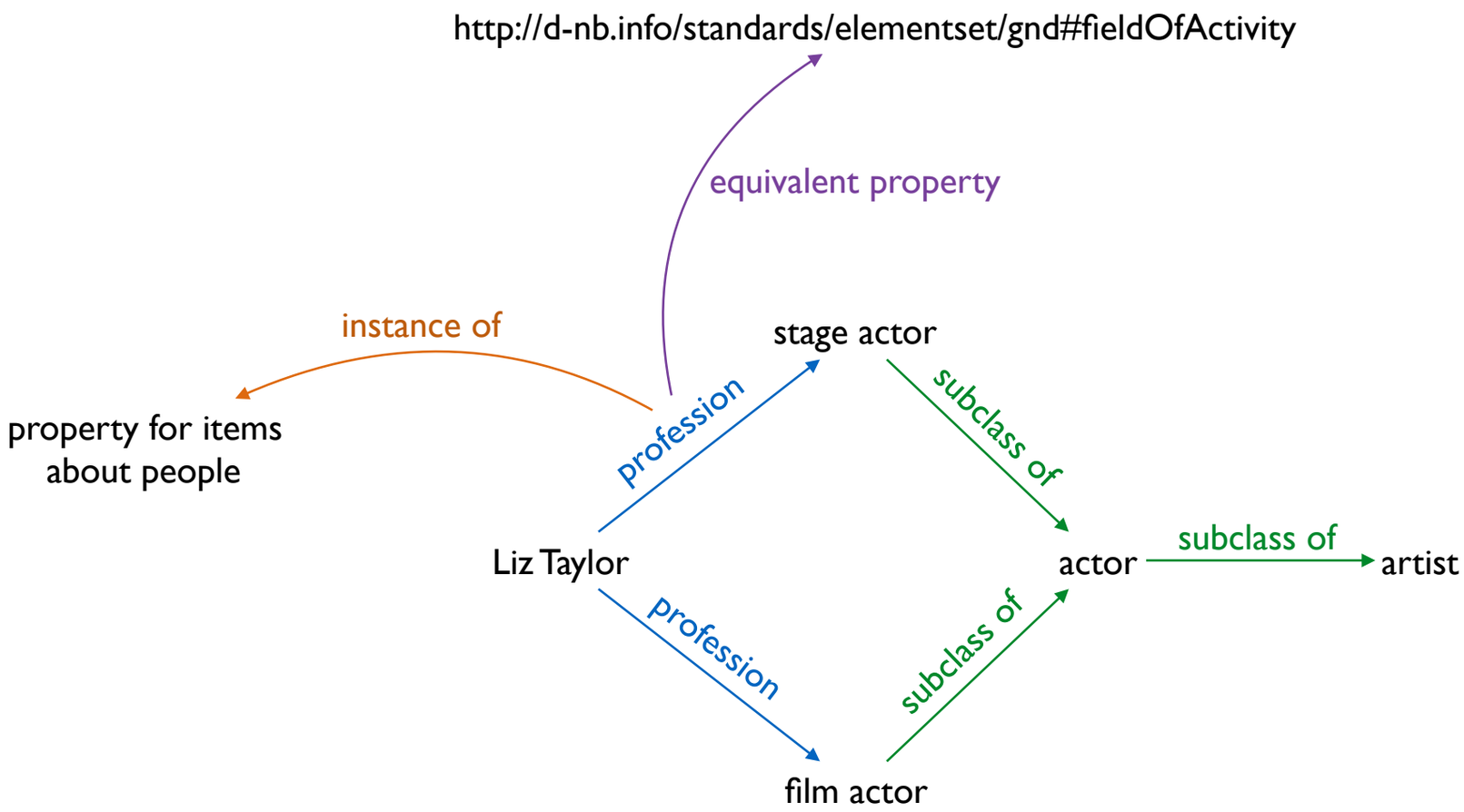



| Profession | |
|---|---|
| Liz Taylor | stage actor |
| Liz Taylor | film actor |

| Subclass of | |
|---|---|
| film actor | actor |
| stage actor | actor |
| actor | artist |

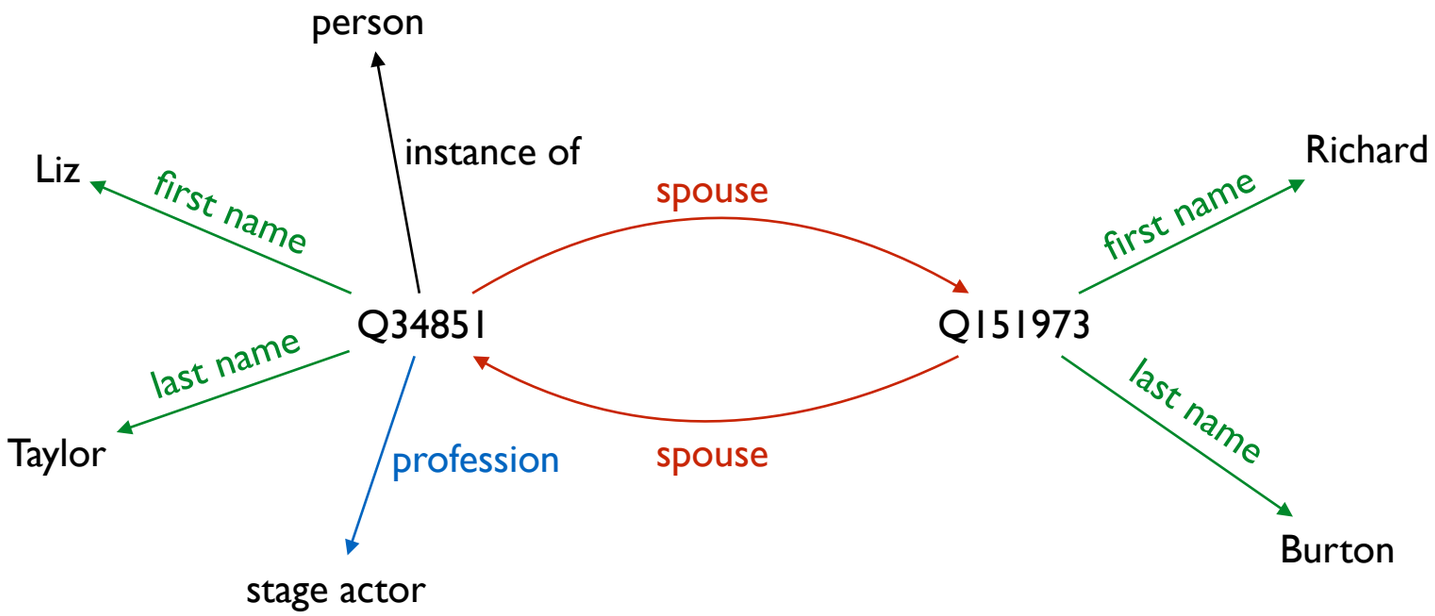
"RDF-like" graph database

# RDF Data Model




"RDF-like" graph database

# What We Consider Today




Edge-labeled, directed graphs

# Graph Database

We assume that $\Sigma$ is a countably infinite set of labels

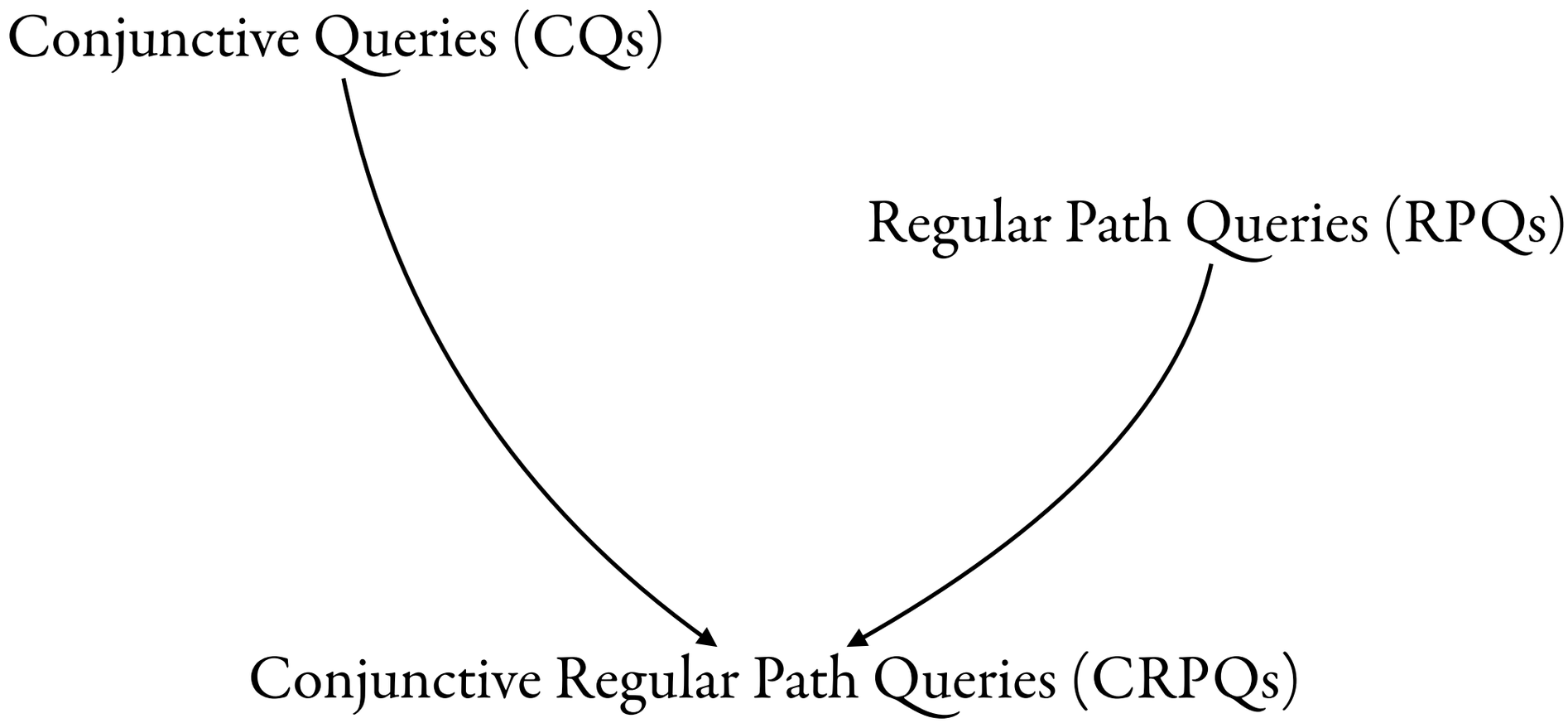
**Definition**

A **graph database** (over $\Sigma$) is a pair $G = (V, E)$ where

- $V$ is a finite set of nodes
- $E \subseteq V \times \Sigma \times V$ is a finite set of edges

# Queries

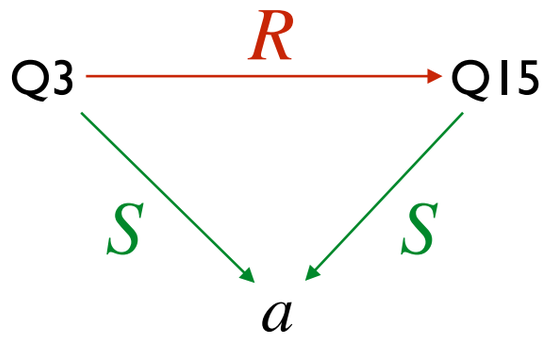
# Plan


Conjunctive Queries (CQs)
Regular Path Queries (RPQs)
Conjunctive Regular Path Queries (CRPQs)

# Conjunctive Queries (CQs)

## Intuition

Not much different
from CQs in relational DBs




## Example (CQ on binary relations)

$R(x, y) \wedge S(x, a) \wedge S(y, a)$

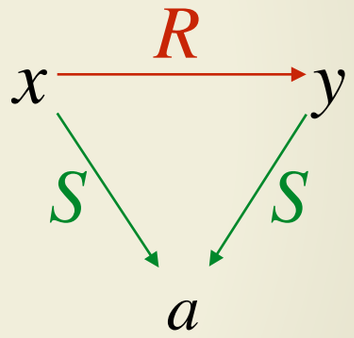
(uses variables $x$, $y$ and constant $a$)

## Example (CQ in graph databases)

More visual notation

$$x \xrightarrow{R} y \;\wedge\; x \xrightarrow{S} a \;\wedge\; y \xrightarrow{S} a$$

or even



# Conjunctive Queries

**Definition (Conjunctive Query over Graphs)**

A **conjunctive query over graphs (CQ)** is an expression of the form

$$\exists \bar{z}\big((x_1 \xrightarrow{a_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{a_n} y_n)\big)$$

where

- $\bar{z}$ is a tuple of variables from $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ and
- $\{a_1, \ldots, a_n\} \subseteq \Sigma$

Main technical difference with CQs over relations:
we only use binary relations here

# Conjunctive Queries

By

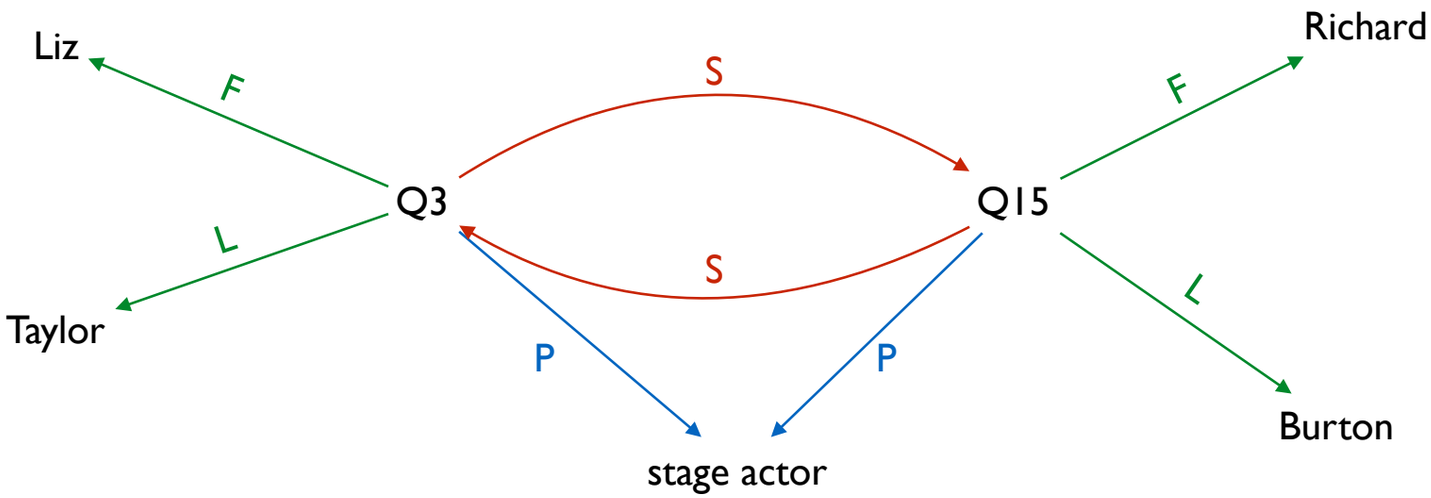
$$Q(\overline{o}) = \exists \overline{z}\Big((x_1 \xrightarrow{a_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{a_n} y_n)\Big)$$

we denote that

$$Q = \exists \overline{z}\Big((x_1 \xrightarrow{a_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{a_n} y_n)\Big)$$

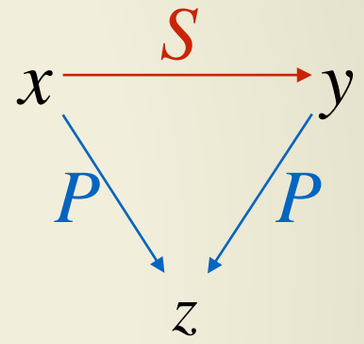
is a conjunctive query and that $\overline{o} \subseteq \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ is the tuple of free variables (or output variables)

# Conjunctive Queries: Example




## Example (CQ on binary relations)

$$Q(x) = (x \xrightarrow{S} y) \wedge (x \xrightarrow{P} z) \wedge (y \xrightarrow{P} z)$$




Returns: {Q3, Q15}

Homomorphism $h_1$: $\{x \mapsto \text{Q3}, y \mapsto \text{Q15}, z \mapsto \text{stage actor}\}$

Homomorphism $h_2$: $\{x \mapsto \text{Q15}, y \mapsto \text{Q3}, z \mapsto \text{stage actor}\}$

# Regular Path Queries

## Why regular path queries?

Conjunctive queries (and even first-order queries) on graphs are limited:

they can only express "local" properties
[Gaifman 1982, Hanf 1965]

Regular path queries overcome this, using regular expressions to query **paths**

## Definition

A **path** in graph $G$ is a sequence

$$p = (v_0, a_1, v_1)\,(v_1, a_2, v_2)\,\ldots\,(v_{n-1}, a_n, v_n)$$

of edges of $G$

# Regular Path Queries

**Definition**

A **regular path query (RPQ)** is an expression of the form

$$x \xrightarrow{r} y$$

where $x$ and $y$ are variables and $r$ is a regular expression over $\Sigma$

(Notice that $r$ can only mention a finite subset of $\Sigma$)

**Semantics**

There are different semantics of RPQs in the literature!

every path

trail

simple path

shortest path

The differences between these are important

# Semantics of RPQs

**Why will we consider these different semantics?**

Each of these semantics is important:

- Every path semantics has been studied most in the literature
- (A variant of) simple path semantics was standard in SPARQL for a while
- Trail semantics is the default in Neo4j Cypher
- Simple path semantics was the first that was studied [Cruz, Mendelzon, Wood 1987]

Members of the OpenCypher project were discussing recently which of the semantics to use for Cypher (www.opencypher.org)

Consensus seems to be: "All should be supported"

# Semantics of RPQs

## Matching Paths

Let $r$ be a regular expression and $G$ be a graph

A path $p = (v_0, a_1, v_1)\ (v_1, a_2, v_2)\ \ldots\ (v_{n-1}, a_n, v_n)$ in $G$ **matches** $r$, if

$$a_1 a_2 \ldots a_n \in L(r)$$

## Semantics of RPQs (every path semantics)

Let $Q = (x \xrightarrow{r} y)$ be a regular path expression and $G$ be a graph

The semantics $[\![Q]\!]_G$ of $Q$ on $G = (V, E)$ is

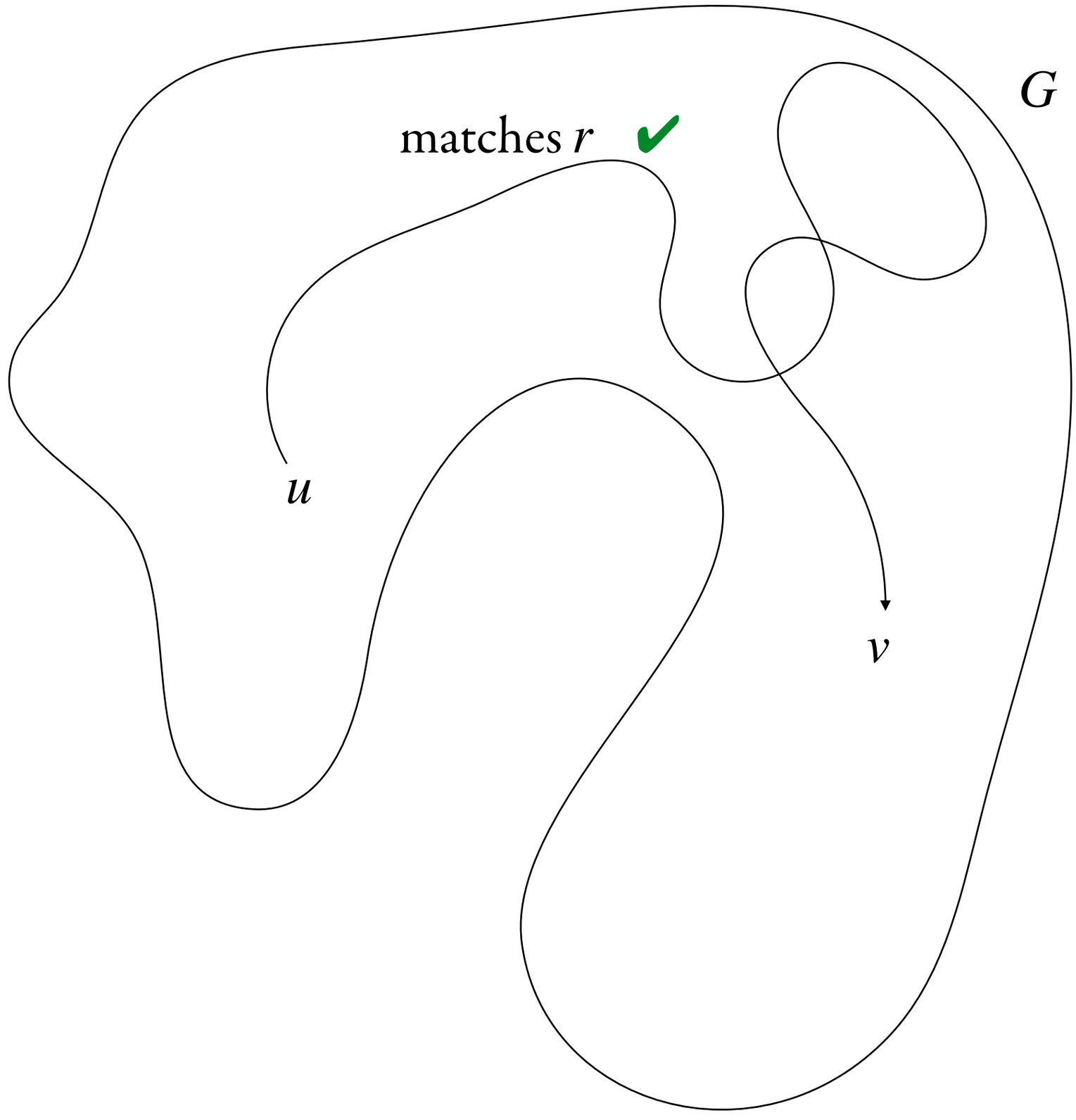
$$[\![Q]\!]_G = \{(u, v) \in V \times V \mid \text{there exists a path } p \text{ from } u \text{ to } v \text{ in } G \text{ that matches } r\}$$

# Semantics of RPQs

RPQ

$Q = (x \xrightarrow{r} y)$

$(u,v)$ is returned iff
there is a path
from $u$ to $v$
that matches $r$



# Semantics of RPQs

## Matching Paths

Let $r$ be a regular expression and $G$ be a graph

A path $p = (v_0, a_1, v_1)\ (v_1, a_2, v_2)\ \dots\ (v_{n-1}, a_n, v_n)$ in $G$ **matches** $r$, if

$$a_1 a_2 \dots a_n \in L(r)$$

## Semantics of RPQs (every path semantics)

Let $Q = (x \xrightarrow{r} y)$ be a regular path expression and $G$ be a graph

The semantics $[\![Q]\!]_G$ of $Q$ on $G = (V, E)$ is

$$[\![Q]\!]_G = \{(u, v) \in V \times V \mid \text{there exists a path } p \text{ from } u \text{ to } v \text{ in } G \text{ that matches } r\}$$

Notice that we do not have any constraint on the path $p$

Hence, "every path" is eligible for the query

## Notation

If $Q = (x \xrightarrow{r} y)$, we sometimes denote $[\![Q]\!]_G$ by $[\![r]\!]_G$

# Simple Paths and Trails

**Definition (Simple path, trail)**

Let $p = (v_0, a_1, v_1)\ (v_1, a_2, v_2)\ \ldots\ (v_{n-1}, a_n, v_n)$ be a path

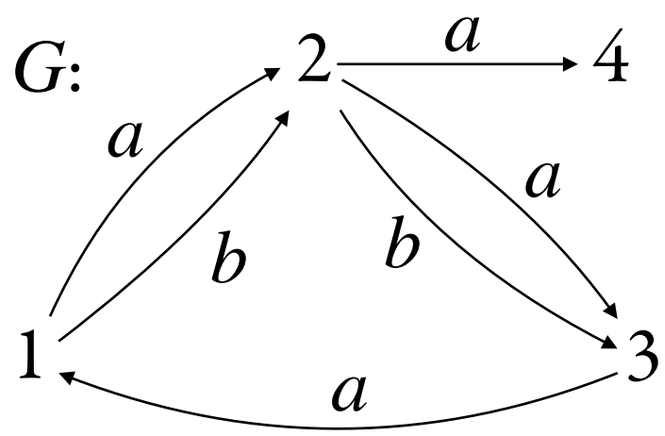
Path $p$ is a **simple path** if

- $v_0$, $v_n$ appear at most once and
- every node in $\{v_1, \ldots, v_{n-1}\}$ appears at most twice in $p$

Path $p$ is a **trail** if

- every edge $(v_{i-1}, a_i, v_i)$ appears at most once in $p$

Examples:



# Semantics of RPQs

**Semantics of RPQs** (simple path semantics)

Let $Q = (x \xrightarrow{r} y)$ be an RPQ and $G$ be a graph

The simple path semantics $[\![Q]\!]^s_G$ of $Q$ on $G = (V, E)$ is

$[\![Q]\!]^s_G = \{(u, v) \in V \times V \mid$ there exists a simple path $p$ from $u$ to $v$ in $G$ that matches $r\}$

# Semantics of RPQs

## Semantics of RPQs (trail semantics)

Let $Q = (x \xrightarrow{r} y)$ be an RPQ and $G$ be a graph

The trail semantics $[\![Q]\!]^t_G$ of $Q$ on $G = (V, E)$ is

$$[\![Q]\!]^t_G = \{(u, v) \in V \times V \mid \text{there exists a trail } p \text{ from } u \text{ to } v \text{ in } G \text{ that matches } r\}$$

# RPQ Semantics: Examples

Take $r = (aa)^*$

then $(1,4) \in [\![r]\!]_G$, $[\![r]\!]^t_G$, and $[\![r]\!]^s_G$

Take $r = (aa)^*a$

then $(1,4) \in [\![r]\!]_G$

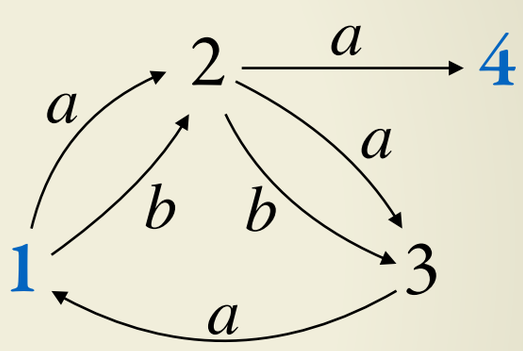
but $(1,4) \notin [\![r]\!]^t_G$ or $[\![r]\!]^s_G$

Take $r = (ab)^*a$

then $(1,4) \in [\![r]\!]_G$ and $[\![r]\!]^t_G$

but $(1,4) \notin [\![r]\!]^s_G$

$G$:



# Conjunctive Regular Path Queries

**Definition (Conjunctive Regular Path Query)**

A conjunctive regular path query (CRPQ) is an expression of the form

$$\exists \bar{z}\Big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\Big)$$

where

- $\bar{z}$ is a tuple of variables from $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ and
- $r_i$ is an RPQ over $\Sigma$ for $i \in [n]$

**Observation**

Since every symbol $a$ in $\Sigma$ is a regular expression,

every CQ over graphs is also a CRPQ

# Conjunctive Regular Path Queries

## Semantics of CRPQs (every path semantics)

Let $Q = \exists \bar{z}\big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\big)$ be a CRPQ and $G = (V, E)$ be a graph

Let $\text{vars}(Q) = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ be the set of variables of $Q$

Then $[\![Q]\!]_G = \{\, \text{h}(\bar{z}) \mid \text{h is a homomorphism from vars}(Q) \text{ to } V$
such that $(\text{h}(x_i), \text{h}(y_i)) \in [\![x_i \xrightarrow{r_i} y_i]\!]_G$ for every $i \in [n]\}$

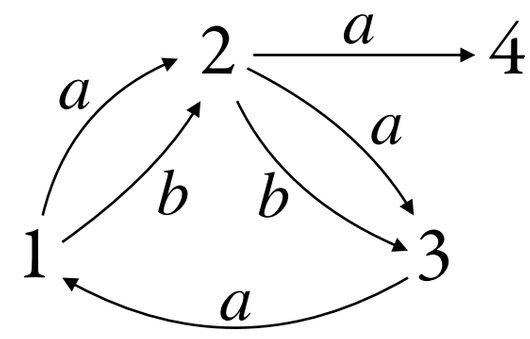
Simple path ($[\![Q]\!]^s_G$) and trail ($[\![Q]\!]^t_G$) semantics for CRPQs are defined analogously: we require that

$(\text{h}(x_i), \text{h}(y_i)) \in [\![x_i \xrightarrow{r_i} y_i]\!]^s_G$ and
$(\text{h}(x_i), \text{h}(y_i)) \in [\![x_i \xrightarrow{r_i} y_i]\!]^t_G$, respectively

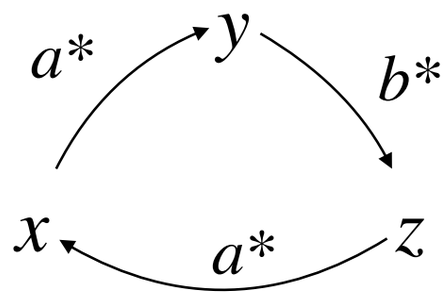
# CRPQs: Examples

$$Q_{aba}(x,y,z) = \left((x \xrightarrow{a^*} y) \wedge (y \xrightarrow{b^*} z) \wedge (z \xrightarrow{a^*} x)\right)$$

$G$:




$Q_{aba}(x,y,z) =$




$$(1,2,3) \in [\![Q_{aba}]\!]_G \,?$$
$$(1,2,2) \in [\![Q_{aba}]\!]_G \,?$$
$$(1,1,1) \in [\![Q_{aba}]\!]_G \,?$$
$$(1,3,1) \in [\![Q_{aba}]\!]_G \,?$$

# Query Evaluation

# Evaluation Problems

**RPQ Evaluation** (every path semantics)

Input: Graph database $G$, pair $(u, v)$ of nodes
regular path query $Q$

Question: Is $(u, v) \in [\![Q]\!]_G$?

**CRPQ Evaluation** (every path semantics)

Input: Graph database G, tuple ū of nodes
conjunctive regular path query $Q$

Question: Is ū $\in [\![Q]\!]_G$ ?

The decision problems for simple path and trail semantics are defined analogously

# Query Evaluation

RPQs

# RPQs, Every Path Semantics

**Theorem**

RPQ Evaluation under every path semantics is in PTIME

**Proof (sketch)**

Let $Q = (x \xrightarrow{r} y)$ be the RPQ, let $G$ be the graph, and $(u,v)$ the pair of nodes

Let $N = (S, A, \delta, I, F)$ be an NFA for $r$

Construct a product $G \times N$, treating $u$ as "initial state" in $G$

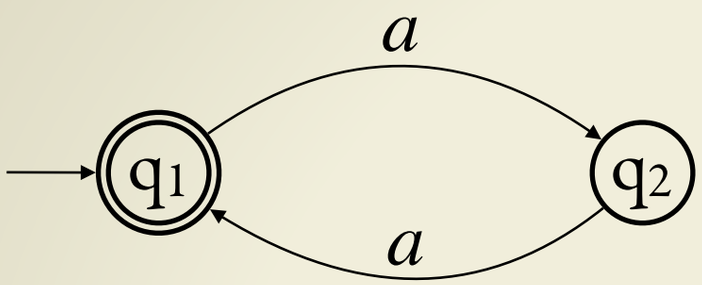
(This is similar to a product between automata)

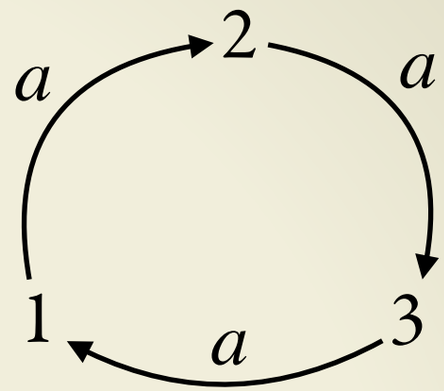
Accept iff there is a path from $(i,u)$ to $(f,v)$ in $G \times N$, for some $i \in I$ and $f \in F$

# Example

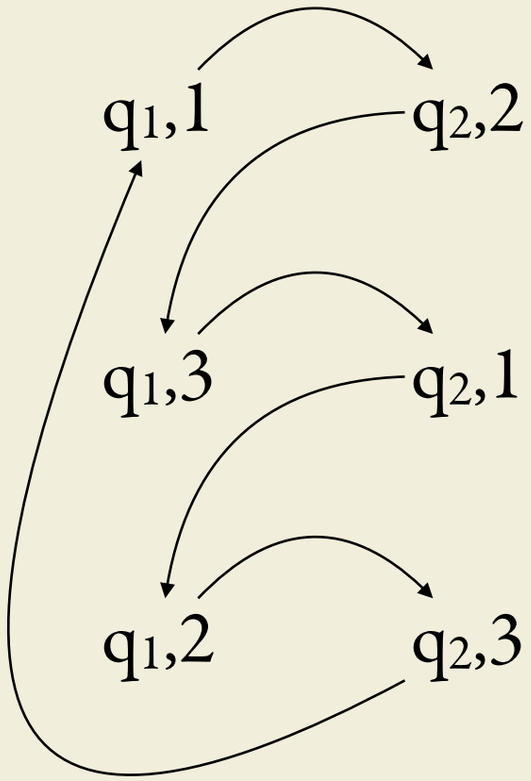
## RPQ Evaluation under Every Path Semantics

Consider the RPQ $r = (aa)^*$




Is (1,2) in $[\![r]\!]_G$ ?



# RPQs, Simple Path Semantics

**Theorem**

RPQ Evaluation under simple path semantics is NP-complete

**Proof (sketch)**

Upper bound:

Guess a path from $u$ to $v$ in $G$ and check if it is simple and matches $r$

Lower bound:

Reduction from Hamiltonian Path

Let $G$ be a directed graph with $n$ nodes and $(u,v)$ a pair of nodes of $G$

Let $G_a$ be obtained from $G$ by labeling each edge with $a$

Then $G$ has a Hamiltonian Path from $u$ to $v$ iff $(u,v)$ in $[\![a^{n-1}]\!]^s_{G_a}$

OK, it's hard

# RPQs, Simple Path Semantics

**Theorem**

RPQ Evaluation under simple path semantics is NP-hard,
even for the RPQ $Q = (x \xrightarrow{(aa)^*} y)$

Reduction from

**Even Length Simple Path**

Given a directed graph $G$ and node pairs $(u,v)$,
is there a simple path of even length from $u$ to $v$?

Even Length Simple Path is NP-complete [Lapaugh, Papadimitriou, Networks 1984]

**Proof (sketch)**

Let $G_a$ be the graph constructed before

Then $G$ has a simple path of even length from $u$ to $v$ iff $(u, v) \in [\![(aa)^*]\!]^s_{G_a}$

# RPQs, Simple Path Semantics

**Theorem**

RPQ Evaluation under simple path semantics is NP-hard,
even for the RPQ $Q = (x \xrightarrow{a^*ba^*} y)$

Reduction from

**Two Disjoint Paths**

Given a directed graph $G$ and node pairs $(u_1,v_1)$ and $(u_2,v_2)$
are there node-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ respectively?

Two Disjoint Paths is NP-complete [Fortune, Hopcroft, Wyllie TCS 1980]

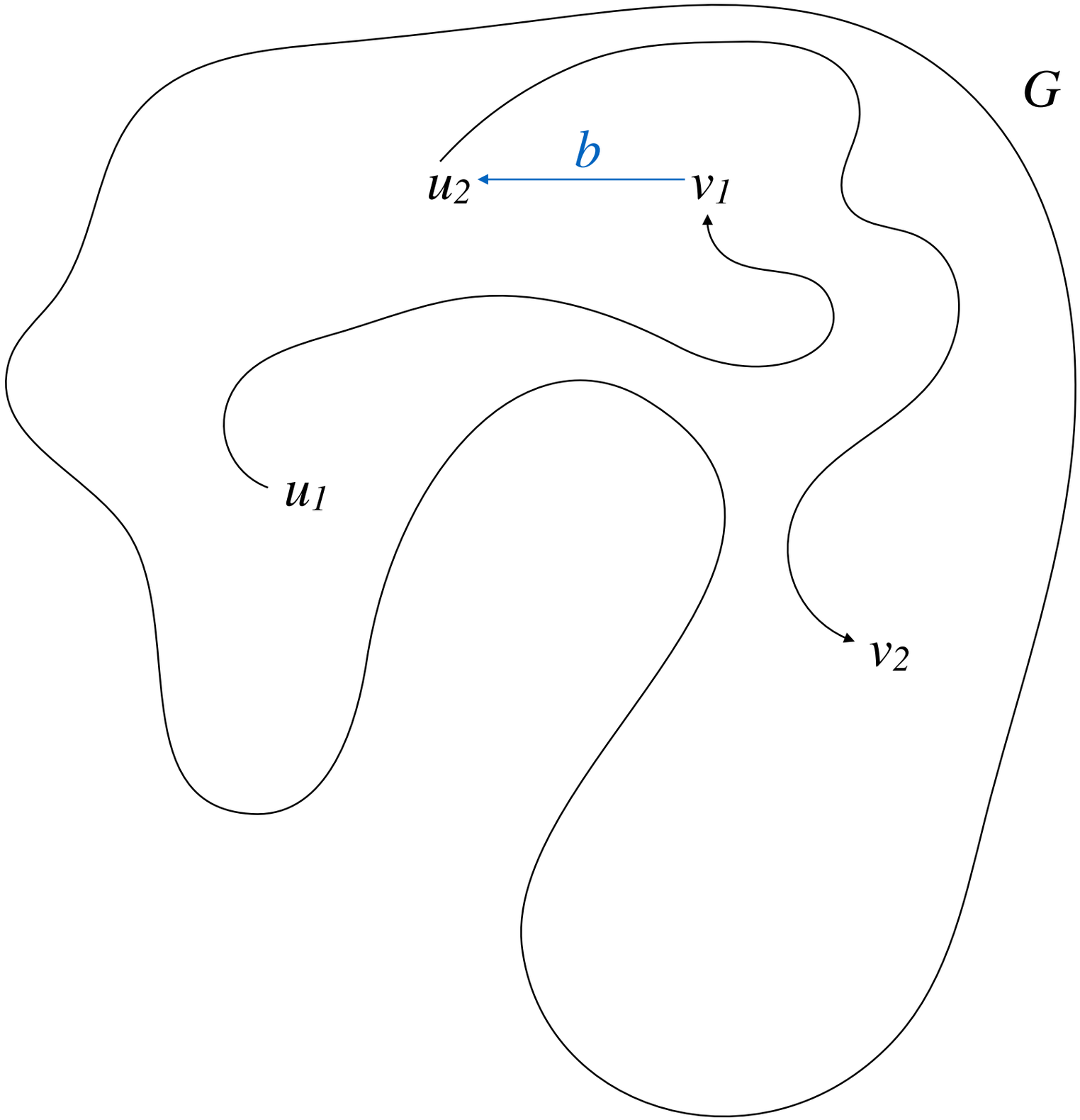
**Proof (sketch)**

Let $G_b$ be obtained from $G_a$ by adding the edge $(v_1, b, u_2)$

Then $G$ has node-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ iff

$$(u_1, v_2) \in [\![a^*ba^*]\!]^s_{G_b}$$

# RPQs, Simple Path Semantics



# RPQs, Trail Semantics

**Theorem**

RPQ Evaluation under trail semantics is NP-hard, even for RPQ $Q = (x \xrightarrow{a^*ba^*} y)$

Reduction from

**Two Edge Disjoint Paths**

Given a directed graph $G$ and node pairs $(u_1,v_1)$ and $(u_2,v_2)$

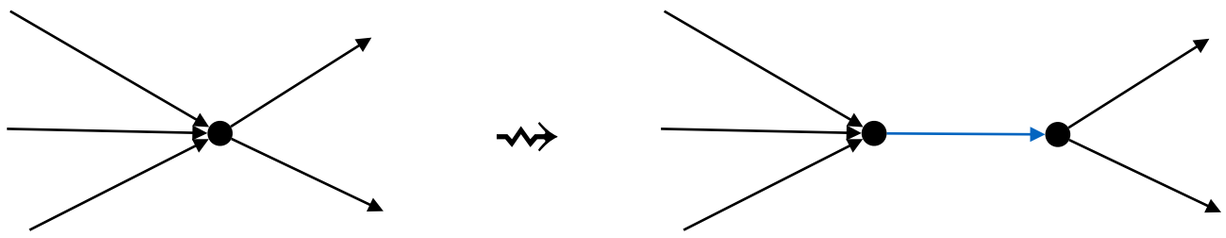
are there edge-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ respectively?

Two Edge Disjoint Paths is NP-complete

Split graph

[LaPaugh, Rivest JCSS 1980]

[Perl, Shiloach JACM 1978]



# RPQs, Trail Semantics

**Theorem**

RPQ Evaluation under trail semantics is NP-hard, even for RPQ $Q = (x \xrightarrow{a^*ba^*} y)$

Reduction from

**Two Edge Disjoint Paths**

Given a directed graph $G$ and node pairs $(u_1,v_1)$ and $(u_2,v_2)$

are there edge-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ respectively?

Two Edge Disjoint Paths is NP-complete

[Fortune, Hopcroft, Wyllie TCS 1980]
[LaPaugh, Rivest JCSS 1980]
[Perl, Shiloach JACM 1978]

**Proof (sketch - same reduction as before)**

Let $G_b$ be obtained from $G_a$ by adding the edge $(v_1, b, u_2)$

Then $G$ has edge-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ iff

$$(u_1, v_2) \in [\![a^*ba^*]\!]^t_{G_b}$$

# RPQs, Trail Semantics

**Theorem**

RPQ Evaluation under trail semantics is NP-hard, even for RPQ $Q = (x \xrightarrow{(aa)^*} y)$

Why?

# Query Evaluation

## CRPQs

# CRPQs, Every Path Semantics

**Theorem**

CRPQ Evaluation under every path semantics is NP-complete

**Proof (sketch)**

Lower bound: immediate from conjunctive queries

Upper bound:
Let $Q = \exists \bar{z}\big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\big)$ be the query

For each regular expression $r_i$, we can compute in polynomial time
a relation $R_i$ containing the tuples $[\![r_i]\!]_G$

Then, evaluation for $Q$ is the same as evaluation of the conjunctive query

$$Q_R = \exists \bar{z}\big(R_1(x_1, y_1) \wedge \cdots \wedge R_n(x_n, y_n)\big)$$

over the relations $R_i$

# CRPQs, Every Path Semantics

Let C be a class of CRPQs

Let $C_{Rel}$ be the class of (relational) CQs, defined as $C_{Rel} = \{Q_R \mid Q \in C\}$

**Corollary**

Let C be a class of CRPQs

Then Evaluation for C under every path semantics is tractable iff
Evaluation for $C_{Rel}$ is tractable in the relational model

# CRPQs, Simple Path / Trail Semantics

**Theorem**

CRPQ Evaluation is NP-complete under simple path and under trail semantics

**Proof (sketch)**

Lower bound: already holds for RPQs
Upper bound: simple guess-and-check algorithm

So, here we don't have a similar corollary
that links to the complexity of CQs over relations

# Overview

| | RPQs | CRPQs |
|---|---|---|
| **every path** | PTIME | NP-complete |
| **simple path** | NP-complete | NP-complete |
| **trail** | NP-complete | NP-complete |

# Query Containment

# Basic Containment Problems

**RPQ Containment**

Input: RPQs $Q_1$ and $Q_2$

Question: Is $[\![Q_1]\!]_G \subseteq [\![Q_2]\!]_G$ for every graph $G$?

**CRPQ Containment**

Input: CRPQs $Q_1$ and $Q_2$

Question: Is $[\![Q_1]\!]_G \subseteq [\![Q_2]\!]_G$ for every graph $G$?

The problems for simple path and trail semantics are analogous

# Query Containment

RPQs

# RPQ Containment

**Theorem**

RPQ Containment is PSPACE-complete

**Proof (sketch)**

Let $Q_1 = (x_1 \xrightarrow{r_1} y_1)$ and $Q_2 = (x_2 \xrightarrow{r_2} y_2)$ be RPQs

It is easy to see that $Q_1 \subseteq Q_2$ iff $L(r_1) \subseteq L(r_2)$

Testing $L(r_1) \subseteq L(r_2)$ for two given regular expressions $r_1$ and $r_2$ is PSPACE-complete

The same proof works for simple path and trail semantics

# Query Containment

## CRPQs

# CRPQ Containment

**Theorem** [Calvanese et al. KR 2000 "The Four Italians"]

CRPQ Containment is EXPSPACE-complete

**Proof (Plan)**

Let $Q_1$ and $Q_2$ be the CRPQs

Upper bound: we reduce the problem to containment of NFAs

We first argue that there exist exponential-size NFAs $A_1$ and $A_2$ such that

$$Q_1 \subseteq Q_2 \quad \text{iff} \quad L(A_1) \subseteq L(A_2)$$

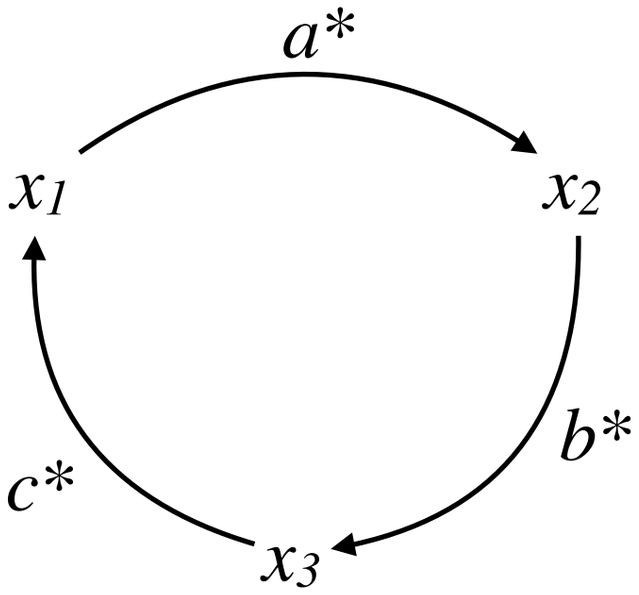
Testing $L(A_1) \subseteq L(A_2)$ can then be done on the fly

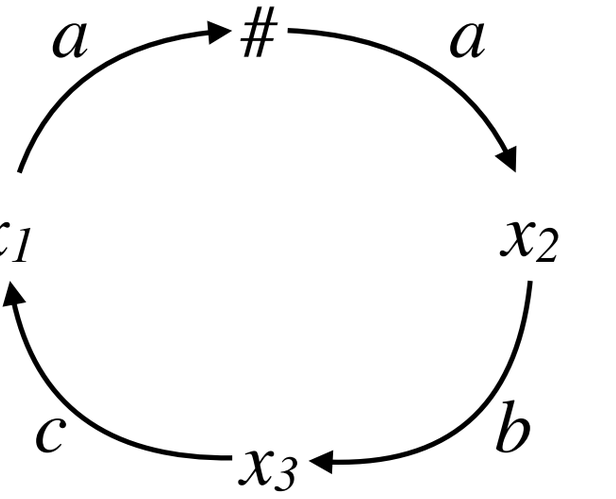
Lower bound: we reduce from Exponential Corridor Tiling

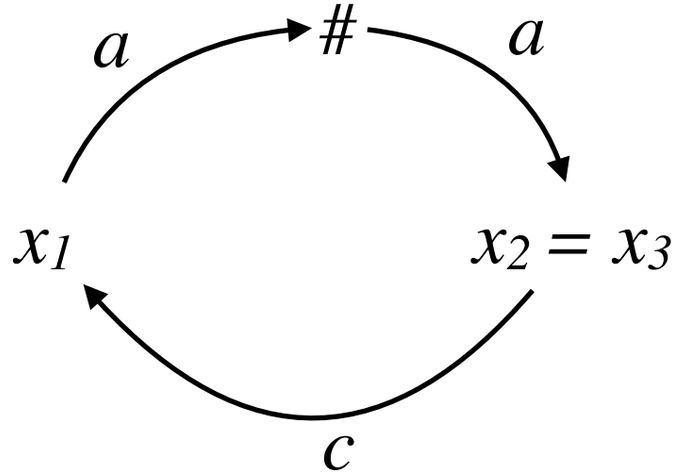
# CRPQ Containment: Upper Bound

Let $Q = \exists \bar{z}\big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\big)$ be a CRPQ

A conjunctive query $Q^e$ is an **expansion of $Q$** if it can be obtained from $Q$ by "replacing each $r_i$ by a path, labeled with a word in $L(r_i)$"



# CRPQ Containment: Upper Bound

Let $Q = \exists \bar{z}\big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\big)$ be a CRPQ

**Definition (Expansion of $Q$)**

A conjunctive query $Q^e$ is an **expansion of $Q$** if there exist words $w_i \in L(r_i)$ such that $Q_e$ can be obtained from $Q$ as follows:

Replace each atom $(x_i \xrightarrow{r_i} y_i)$

- by $(x_i = y_i)$ if $w_i = \varepsilon$
- by a conjunction $(x_i \xrightarrow{a_1} \#_i^1) \wedge (\#_i^1 \xrightarrow{a_2} \#_i^2) \wedge \cdots \wedge (\#_i^{k_i} \xrightarrow{a_{k_i}} y_i)$ such that $w_i = a_1 \cdots a_{k_i}$

We assume that all variables $\#_i^j$ are new and pairwise distinct

**Observation**

There is always a homomorphism from $Q$ to $Q^e$, namely the identity

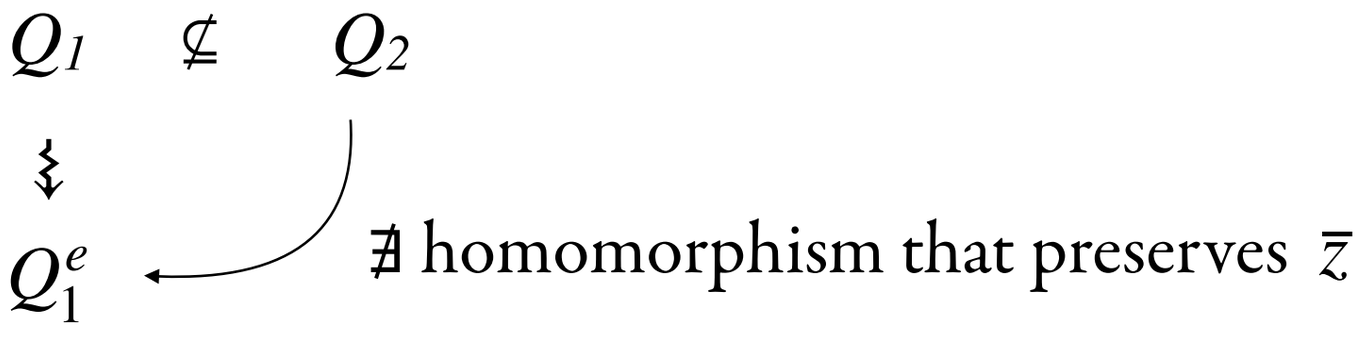
# CRPQ Containment: Upper Bound

Let $Q_1$ and $Q_2$ be CRPQs

We assume w.l.o.g. that $Q_1$ and $Q_2$ have the same free variables $\bar{z}$
and all other variables are disjoint

**Lemma [Calvanese et al. 2000]**

$Q_1 \not\subseteq Q_2$ iff there exists an expansion $Q_1^e$ of $Q_1$,
for which there is no homomorphism
from $Q_2$ to $Q_1^e$ that is the identity on $\bar{z}$




This is what we will try to test with automata $A_1$ and $A_2$

# CRPQ Containment: Upper Bound

Let $Q_1 = \exists \bar{z}\big((x_1 \xrightarrow{r_1} y_1) \wedge \cdots \wedge (x_n \xrightarrow{r_n} y_n)\big)$

We can encode expansions of $Q_1$ as words

$\$\ x_1 w_1 y_1\ \$\ x_2 w_2 y_2\ \$\ \ldots\ \$\ x_n w_n y_n\ \$$

over the alphabet $\Sigma \cup \mathrm{Vars}(Q) \cup \{\$,\#\}$

Intuition:

- each $x_i \xrightarrow{r_i} y_i$ corresponds to $x_i w_i y_i$
- each word $w_i$ is of the form $a_1\ \#\ a_2\ \#\ \ldots\ \#\ a_{ki}$ where $a_1 \ldots a_{ki}$ in $L(r_i)$
- each $x_i$, $y_i$ and # can be seen as a variable in the expansion

**Exercise**

Given $Q_1$, show that there is a polynomial size NFA that checks if a given word $w$ encodes an expansion of $Q_1$

# CRPQ Containment: Upper Bound

We can also encode expansions of $Q_1$ as words

$$\$\ X_1 w_1 Y_1\ \$\ X_2 w_2 Y_2\ \$\ \ldots\ \$\ X_n w_n Y_n\ \$$$

over the alphabet $\Sigma \cup 2^{\mathrm{Vars}(Q)} \cup \{\$,\#\}$

Here the $X_i$ and $Y_i$ are sets of variables

The idea is that

(1) word $w_i$ is in $L^{\#}(r_i)$ for all $i \in [n]$ → $a_1 \# a_2 \# \ldots \# a_k$ for $a_1 \ldots a_k$ in $L(r_i)$

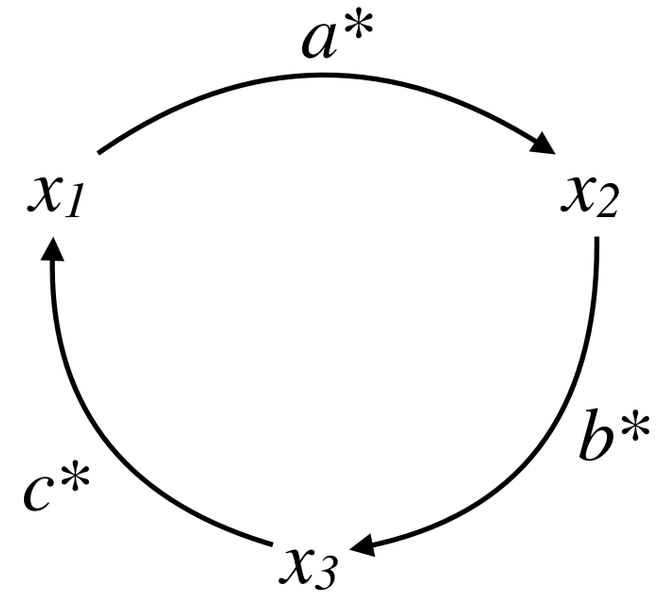
(2) $x_i \in X_i$ and $y_i \in Y_i$ for all $i \in [n]$

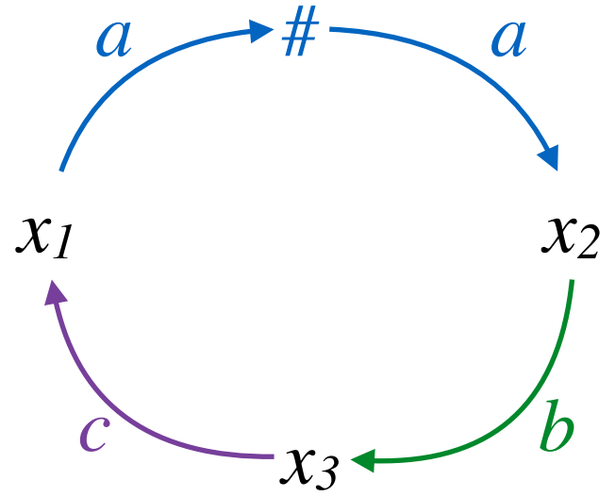
(3) the sets $X_i$, $Y_i$ form a partition of $\mathrm{Vars}(Q)$ (but sets are allowed to repeat!)

(4) whenever $w_i = \varepsilon$, then $X_i = Y_i$

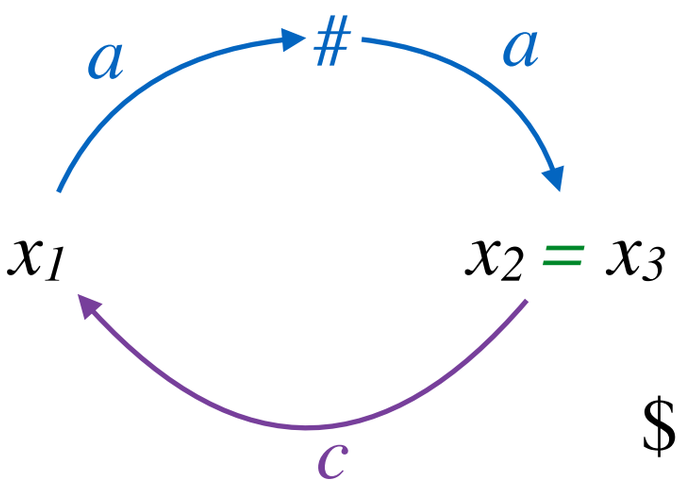
Such words are called **$Q_1$-words**

# Intermezzo: Examples




$ {$x_1$} $a\#a$ {$x_2$} $ {$x_2$} $b$ {$x_3$} $ {$x_3$} $c$ {$x_1$} $




$ {$x_1$} $a\#a$ {$x_2, x_3$} $ {$x_2, x_3$} {$x_2, x_3$} $ {$x_2, x_3$} $c$ {$x_1$} $

# CRPQ Containment: Upper Bound

We can also encode expansions of $Q_1$ as words

$$\$\ X_1 w_1 Y_1\ \$\ X_2 w_2 Y_2\ \$\ \ldots\ \$\ X_n w_n Y_n\ \$$$

over the alphabet $\Sigma \cup 2^{\mathrm{Vars}(Q)} \cup \{\$,\#\}$

Here the $X_i$ and $Y_i$ are sets of variables

The idea is that

(1) word $w_i$ is in $L^{\#}(r_i)$ for all $i \in [n]$ → $a_1 \# a_2 \# \ldots \# a_k$ for $a_1 \ldots a_k$ in $L(r_i)$

(2) $x_i \in X_i$ and $y_i \in Y_i$ for all $i \in [n]$

(3) the sets $X_i$, $Y_i$ form a partition of $\mathrm{Vars}(Q)$ (but sets are allowed to repeat!)

(4) whenever $w_i = \varepsilon$, then $X_i = Y_i$

Such words are called **$Q_1$-words**

**Can we recognize $Q_1$-words with an automaton $A_1$?**

(1) Polynomial size NFA $A_{1,1}$

(2) Polynomial size NFA $A_{1,2}$

(3) Exponential size NFA $A_{1,3}$

(4) Exponential size NFA $A_{1,4}$

} Use $A_{1,1} \times A_{1,2} \times A_{1,3} \times A_{1,4} \times A_{wf}$ where $A_{wf}$ tests well-formedness

# CRPQ Containment: Upper Bound

We now want to define $A_2$

We first think about "annotated $Q_1$-words", i.e., words of the form

$$\$ (\ell_1,\gamma_1) \ \dots \ (\ell_m,\gamma_m) \$$$

where $\$\ell_1 \dots \ell_m \$$ is a $Q_1$-word and $\gamma_i \subseteq \mathrm{Vars}(Q_2)$ for all $i$

**Intuition**

The variables in $\gamma_i$ are mapped to the node $\ell_i$ if $\ell_i \subseteq \mathrm{Vars}(Q_1) \cup \{\#\}$

We now want to see:

Can an automaton $A'_2$ test if an annotated $Q_1$-word $W_a$ encodes

a $Q_1$-expansion $Q_e$

such that $Q_2$ returns the same answer as $Q_1$ on $Q_e$?

# CRPQ Containment: Upper Bound

We have annotated $Q_1$-word \$ $(\ell_1,\gamma_1)$ ... $(\ell_m,\gamma_m)$ \$ with $Q_1$-word W = \$ $\ell_1$ ... $\ell_m$ \$

**Automaton $A'_2$ tests**

(1) for every $\ell_i \subseteq \mathrm{Vars}(Q_1)$ containing an output variable z, every occurrence $\ell_j$ of $\ell_i$ is annotated with a set $\gamma_j$ that contains z

(2) if a variable $y \in \mathrm{Vars}(Q_2)$ appears in $(\ell_i,\gamma_i)$, then either:

- $\ell_i$ = # and $y$ only appears in $\gamma_i$, or
- $\ell_i \subseteq \mathrm{Vars}(Q_1)$ and $y$ appears in every $\gamma_j$ for which $\ell_i = \ell_j$

(3) for every conjunct $(x'_i \xrightarrow{r'_i} y'_i)$ of $Q_2$, whether the path from $x'_i$ to $y'_i$ matches $r'_i$

How can this be done?

(1) Exponential size NFA

(2) Exponential size NFA

(3) Polynomial size two-way NFA ⇝ exponential size NFA

Automaton $A_2$ reads a $Q_1$-word W, guesses the annotations, and simulates (1)-(3)

# CRPQ Containment: Lower Bound

We reduce from exponential corridor tiling

**Definition (Exponential Corridor Tiling)**

A tiling system is a tuple $\mathbf{T} = (T, H, V, t_s, t_f, n)$ where

- $T$ is a finite set of tile types
- $\mathrm{H} \subseteq T \times T$ is the set of horizontal constraints
- $V \subseteq T \times T$ is the set of vertical constraints
- $t_s \in \mathrm{T}$ is the start tile
- $t_f \in \mathrm{T}$ is the finish tile
- $n \in \mathbb{N}$

# CRPQ Containment: Lower Bound

**Definition (Exponential Corridor Tiling)**

Let $\mathrm{T} = (T, H, V, t_s, t_f, n)$ be a tiling system

It has an exponential corridor solution if there exists an $m \in \mathbb{N}$ and mapping

$$\text{bathroom} : [2^n] \times [m] \to T$$

such that

- the start tile type is correct: $\text{bathroom}(1,1) = t_s$
- the finishing tile type is correct: $\text{bathroom}(2^n, m) = t_f$
- the horizontal constraints are correct: $(\text{bathroom}(x,y), \text{bathroom}(x+1,y))$ in $H$
- the vertical constraints are correct: $(\text{bathroom}(x,y), \text{bathroom}(x,y+1))$ in $V$

**Theorem**

Deciding if a tiling system has an exponential corridor solution is EXPSPACE-complete

# CRPQ Containment: Lower Bound

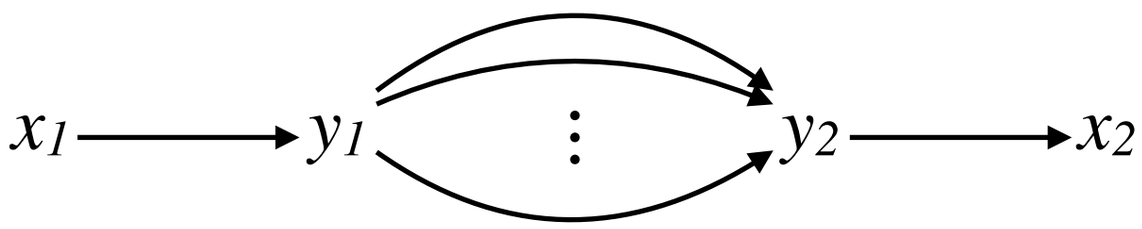
Let $\mathbf{T} = (T, H, V, t_s, t_f, n)$ be an instance of exponential tiling

Plan: define queries $Q_1$ and $Q_2$ such that $Q_1 \not\subseteq Q_2$ iff $\mathbf{T}$ does not have a valid tiling,
i.e. every tiling has some error

$Q_1(x_1, x_2) = (x_1 \xrightarrow{r} x_2)$ with $r = 0^n\ t_s\ ((0+1)^n\ T)^*\ 1^n\ t_f$

$$Q_2(x_1, x_2) = (x_1 \xrightarrow{r_{pre}} y_1) \wedge \left(\bigwedge_{i=0}^{n} y_1 \xrightarrow{r_i} y_2\right) \wedge (y_2 \xrightarrow{r_{suff}} x_2)$$

with $r_{pre} = ((0+1)^n\ T)^*$ and $r_{suff} = ((0+1)^n\ T)^*$



# CRPQ Containment: Lower Bound

$$Q_2(x_1, x_2) = (x_1 \xrightarrow{r_{pre}} y_1) \wedge \left(\bigwedge_{i=0}^{n} y_1 \xrightarrow{r_i} y_2\right) \wedge (y_2 \xrightarrow{r_{suff}} x_2)$$

$$r_i = r_H + r_{Vi} + r_c$$

- $r_c$: counter error
- $r_{Vi}$: vertical error
- $r_H$: horizontal error

$$r_H = \sum_{(t_1,t_2)\notin H} ((0+1)^n\ T)^*\ (0+1)^n\ t_1\ (0+1)^n\ t_2\ ((0+1)^n\ T)^*$$

# CRPQ Containment: Lower Bound

$$Q_2(x_1, x_2) = (x_1 \xrightarrow{r_{pre}} y_1) \wedge \Big(\bigwedge_{i=0}^{n} y_1 \xrightarrow{r_i} y_2\Big) \wedge (y_2 \xrightarrow{r_{suff}} x_2)$$

$$r_i = r_H + r_{Vi} + r_c$$

- $r_c$: counter error
- $r_{Vi}$: vertical error
- $r_H$: horizontal error

$$r_{V0} = \sum_{(t_1,t_2)\notin V} (0+1)^n\ t_1\ ((0+1)^n\ T)^*\ (0+1)\ t_2$$

$$r_{Vi} = r^0_{Vi} + r^1_{Vi} \quad \text{for } i > 0, \text{ with} \quad r^b_{Vi} = (0+1)^{i-1}\ b\ (0+1)^{n-i}\ T\ ((0+1)^*\ b\ (0+1)^*\ T)^*\ \overline{b}^n\ T\ ((0+1)^*\ b\ (0+1)^*\ T)^*\ (0+1)^{i-1}\ b\ (0+1)^{n-i}\ T$$

$\overline{b} = 1 - b$

**Exercise**

Define $r_c$, which should match consecutive $(0+1)^n$-blocks that don't encode consecutive binary numbers

# CRPQ Containment

This concludes the proof!

**Theorem** [Calvanese et al. KR 2000 "The Four Italians"]

CRPQ Containment is EXPSPACE-complete

Actually, the original proof also shows the result for conjunctive two-way regular path queries

# Trees versus Graphs

"Those who don't learn from history ..."

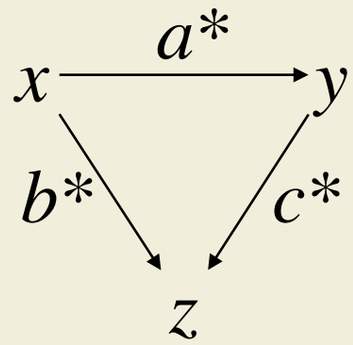
... risk having their papers rejected by the old folks

# But What About Acyclic CRPQs?

Let's call a CRPQ **acyclic**(*) if its associated graph is a tree (ignoring edge directions)

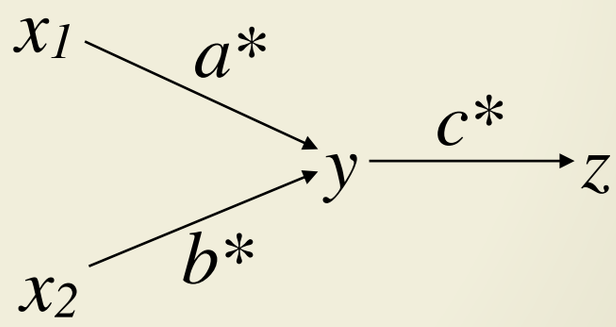
## Example

$$x \xrightarrow{a^*} y \;\wedge\; x \xrightarrow{b^*} z \;\wedge\; y \xrightarrow{c^*} z$$




✗

$$x_1 \xrightarrow{a^*} y \;\wedge\; x_2 \xrightarrow{b^*} y \;\wedge\; y \xrightarrow{c^*} z$$




✔

(*) for the sake of simplicity -- "real" acyclicity should also include forests

# But What About Acyclic CRPQs?

Let's call a CRPQ **acyclic**(*) if its associated graph is a tree

For the sake of simplicity, let's only consider Boolean CRPQs

Denote by $[\![Q]\!]_{\mathbf{G}}$ the set of graphs on which $Q$ is satisfied

Denote by $[\![Q]\!]_{\mathbf{T}}$ the set of trees on which $Q$ is satisfied

**Important observation**

If $Q_1$ and $Q_2$ are acyclic, then $[\![Q_1]\!]_{\mathbf{G}} \subseteq [\![Q_2]\!]_{\mathbf{G}}$ iff $[\![Q_1]\!]_{\mathbf{T}} \subseteq [\![Q_2]\!]_{\mathbf{T}}$

Intuition: A counterexample graph can be unfolded to a tree

(*) for the sake of simplicity -- "real" acyclicity should also include forests

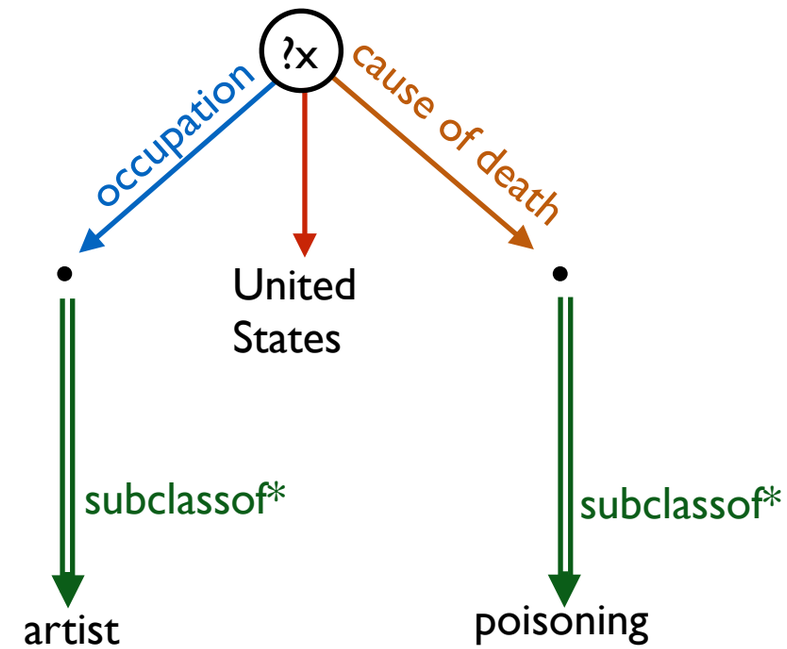
# But What About Acyclic CRPQs?

Papers where this argument has been made (almost certainly incomplete):

[Miklau, Suciu, JACM 2004]
[Reutter, CoRR 2013]
[Barcelo, Perez, Reutter, AMW 2013]
[Czerwiński, M., Niewerth, Parys, JACM 2018]

# What Does This Mean?

If you have queries that behave like




- node labels
- edge labels
- transitive closures
- (wildcards)

...then you can use results from tree patterns on XML data:

**Theorem [Miklau, Suciu JACM 2004]**

Containment of tree patterns is coNP-complete

**Theorem [Czerwinski et al. JACM 2018]**

Minimization of tree patterns is $\Sigma_2^P$-complete

(and minimization ≠ deleting edges)

...and much, much more!

# Data Values

# Queries With Data Value Comparisons

Until now, we never compared labels with each other

Example:

- Return pairs of people with the same last name

This idea leads to different types of queries, e.g., adding conjuncts

$x \sim y$ or $x \not\sim y$

satisfied if nodes $x$ and $y$ have the same, resp., different value

Such queries are usually considered on a different data model

(data words, data trees, data graphs)

but since we chose $\Sigma$ infinite, the main argument also works here

# Queries With Data Value Comparisions

Consider the query $L_{eq}$, matching all paths that contain two equal values

Let $\overline{L_{eq}}$ be its complement,

matching all paths containing pairwise different values

**Theorem**

Evaluation of $\overline{L_{eq}}$ on graph databases is NP-complete

**Proof (sketch)**

Reduction from Edge-Disjoint Paths

Let $(G, u_1, v_1, u_2, v_2)$ be an instance of edge-disjoint paths

Let $G'$ be obtained from $G$ by giving each edge a unique label,

copying the entire graph, obtaining $G_1$ and $G_2$, and adding the edge $(v_1, u_2)$

Then $G$ has edge-disjoint paths $p_1$ and $p_2$, from $u_1$ to $v_1$ and from $u_2$ to $v_2$ iff $G'$ has a path from $u_1$ to $v_2$ matching $\overline{L_{eq}}$

The $\overline{L_{eq}}$ problem can be circumvented by going to XPath-like languages on graphs

[Libkin et al. JACM 16]

# Real Queries

# How Do Real RPQs Look Like?

## Infinite languages

| Expression Type | Relative |
|---|---|
| $A^*$ | 29.10 % |
| $a^*$ | 19.66 % |
| $a^*b$ | 7.73 % |
| $a^+$ | 1.54 % |
| $A^+$ | |
| $(ab^*)+c$ | |
| $a^*b?$ | |
| $abc^*$ | |
| $a^*+b$ | |
| $a+b^+$ | |
| $a^+ + b^+$ | |
| $(ab)^*$ | |

Empty cells are < 0.01%

## Finite languages

| Expression Type | Relative |
|---|---|
| $A$ | 32.10 % |
| $a_1 \dots a_k$ | 8.66 % |
| $a_1? \dots a_k?$ | 1.15 % |
| $aA?$ | 0.01 % |
| $a_1\, a_2? \dots a_k?$ | 0.01 % |
| $A_1 \dots A_k$ | |
| $A?$ | |

$A$, $A_i$: Set of symbols

$a,b,c,a_i$ : Symbols

~250K RPQs
in 56 M unique queries

# How Do Real RPQs Look Like?

## Infinite languages

| Expression Type | Relative |
|---|---|
| a* | 50.48 % |
| a*b | 17.07 % |
| ab*c* | 1.49 % |
| A* | 0.60 % |
| ab*c | 0.22 % |
| a*b* | 0.11 % |
| abc* | 0.05 % |
| a?b* | 0.03 % |
| A+ | 0.01 % |
| Ab* | |
| other | |

Empty cells are < 0.01%

## Finite languages

| Expression Type | Relative |
|---|---|
| $a_1$ ... $a_k$ | 24.26 % |
| A | 5.52 % |
| A? | 0.06 % |
| $a_1$ $a_2$? ... $a_k$? | 0.05 % |
| ^a | 0.04 % |
| abc? | 0.01 % |
| other | |

A, $A_i$: Set of symbols

a,b,c,$a_i$ : Symbols

~55M RPQs
in 207 M robotic queries

# Almost All Expressions are Simple

**Definition (Simple Transitive Expression)**

An atomic expression is a disjunction $(a_1 + \ldots + a_n)$ of symbols
We denote atomic expressions by A

A local expression is a concatenation of the form

$A_1 \ldots A_k$ or $A_1? \ldots A_k?$

"follow a path of length k" "follow a path of length at most k"

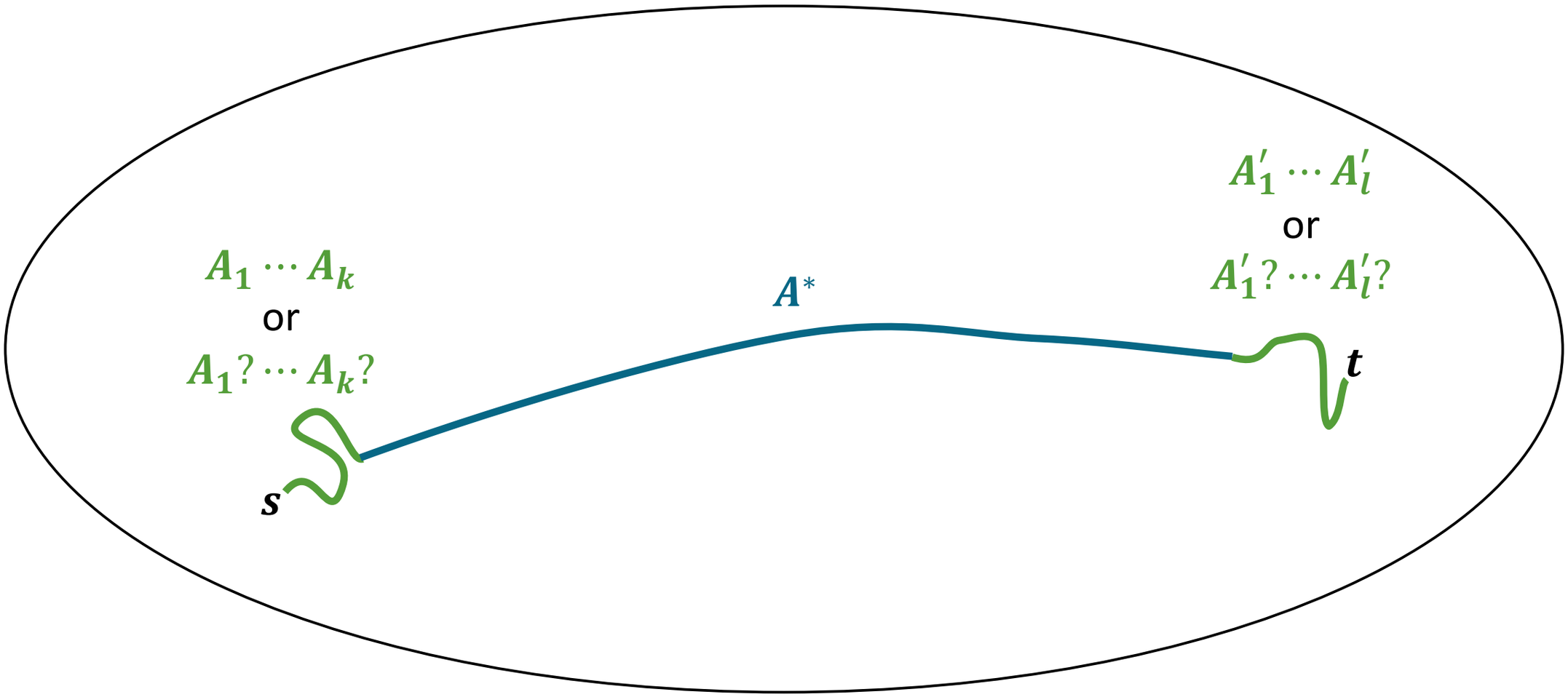
A simple transitive expression (STE) is of the form

$$L_1\ A^*\ L_2$$

where $L_1$ and $L_2$ are local expressions

Here we allow $A = \varnothing$ to express some finite languages

# Simple Transitive Expressions




**Simple Transitive Expression**

$L_1\ A^*\ L_2$ where $L_1$ and $L_2$ are local expressions

# How Do Real RPQs Look Like?

## Infinite languages

| Expression Type | Relative |
|---|---|
| $A^*$ | 29.10 % |
| $a^*$ | 19.66 % |
| $a^*b$ | 7.73 % |
| $a^+$ | 1.54 % |
| $A^+$ | |
| $(ab^*)+c$ | |
| $a^*b?$ | |
| $abc^*$ | |
| $a^*+b$ | |
| $a+b^+$ | |
| $a^+ + b^+$ | |
| $(ab)^*$ | |

## Finite languages

| Expression Type | Relative |
|---|---|
| $A$ | 32.10 % |
| $a_1 \dots a_k$ | 8.66 % |
| $a_1? \dots a_k?$ | 1.15 % |
| $aA?$ | 0.01 % |
| $a_1\, a_2? \dots a_k?$ | 0.01 % |
| $A_1 \dots A_k$ | |
| $A?$ | |

99.99% are STEs

# How Do Real RPQs Look Like?

## Infinite languages

| Expression Type | Relative |
|---|---|
| $a^*$ | 50.48 % |
| $a^*b$ | 17.07 % |
| $ab^*c^*$ | 1.49 % |
| $A^*$ | 0.60 % |
| $ab^*c$ | 0.22 % |
| $a^*b^*$ | 0.11 % |
| $abc^*$ | 0.05 % |
| $a?b^*$ | 0.03 % |
| $A^+$ | 0.01 % |
| $Ab^*$ | |
| other | |

## Finite languages

| Expression Type | Relative |
|---|---|
| $a_1 \dots a_k$ | 24.26 % |
| $A$ | 5.52 % |
| $A?$ | 0.06 % |
| $a_1 a_2? \dots a_k?$ | 0.05 % |
| $\hat{}a$ | 0.04 % |
| $abc?$ | 0.01 % |
| other | |

98.40% are STEs

# Why Am I Saying This?

You can use this to prove theorems!

**RPQ Evaluation** (simple path semantics)

Input: Graph database $G$, pair $(u, v)$ of nodes
regular path query $Q$

Question: Is $(u, v) \in [\![Q]\!]^s_G$?

Is this still NP-complete for STEs?

Yes, take the reduction from Hamilton Path from before

But what if we take a closer look?

# Why Am I Saying This?

You can use this to prove theorems!

**RPQ Evaluation for R** (simple path semantics)

Input: Graph database $G$, pair $(u, v)$ of nodes, regular path query $Q$ from **R**

Question: Is $(u, v) \in [\![Q]\!]^s_G$?

Example classes **R**:

$\underbrace{aa \ldots a}_{k}$ for $k$ in $\mathbb{N}$ denote this by $a^k$

$aa \ldots aa^*$ for $k$ in $\mathbb{N}$ denote this by $a^k a^*$

**Theorem [Alon, Yuster, Zwick, JACM 1995]**

Evaluation for $a^k$ under simple path semantics is in FPT

Color coding technique

**Theorem [Fomin et al., JACM 2016]**

Evaluation for $a^k a^*$ under simple path semantics is in FPT

Representative sets technique

# Why Am I Saying This?

You can use this to prove theorems!

**RPQ Evaluation for R** (simple path semantics)

Input: Graph database $G$, pair $(u, v)$ of nodes, regular path query $Q$ from **R**

Question: Is $(u, v) \in [\![Q]\!]^s_G$?

**Theorem [M., Trautner, ICDT'18]**

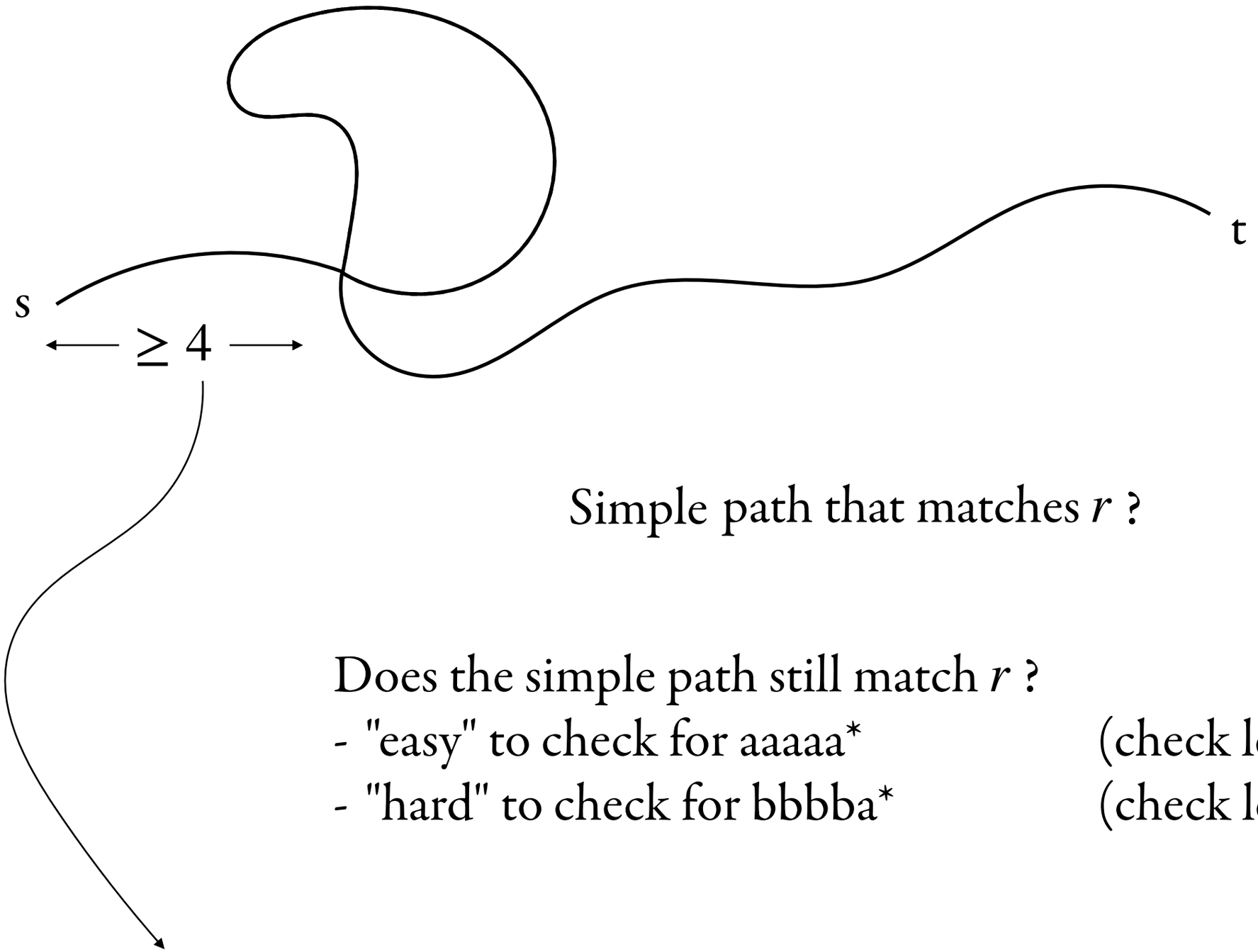
Let **R** be a class(*) of STEs

if **R** is cuttable, then Evaluation for R under simple path semantics is FPT

otherwise, Evaluation for **R** under simple path semantics is W[1]-hard

(*) satisfying a mild condition, needed for the hardness proof

# Cuttability




Simple path that matches $r$ ?

Does the simple path still match $r$ ?
- "easy" to check for aaaaa* (check length)
- "hard" to check for bbbba* (check length + label)

cut border for bbbba*

# Why Am I Saying This?

You can use this to prove theorems!

**RPQ Evaluation for R** (simple path semantics)

Input: Graph database $G$, pair $(u, v)$ of nodes, regular path query $Q$ from **R**

Question: Is $(u, v) \in [\![Q]\!]^s_G$?

**Theorem [M., Trautner, ICDT'18]**

Let **R** be a class(*) of STEs

if **R** is cuttable, then Evaluation for R under simple path semantics is FPT

otherwise, Evaluation for **R** under simple path semantics is W[1]-hard

(*) satisfying a mild condition, needed for the hardness proof

# Concluding Remarks

# Concluding Remarks

## What Have We Done?

- Looked at the most studied query formalisms for graph databases:
  RPQs and CRPQs
  - There is more: C2RPQs, UCRPQs, UC2RPQs, ...
- We studied their most important decision problems:
  Evaluation and Containment
- We did brief excursions to tree-structures and data values
  - Both lead to an entire world of exciting research problems
- We showed that investigating actual queries can open exciting new perspectives on research problems

Graph Data Management is an exciting research direction,
with plenty of theory questions
and plenty of interest from industry about our results

# Thank You!