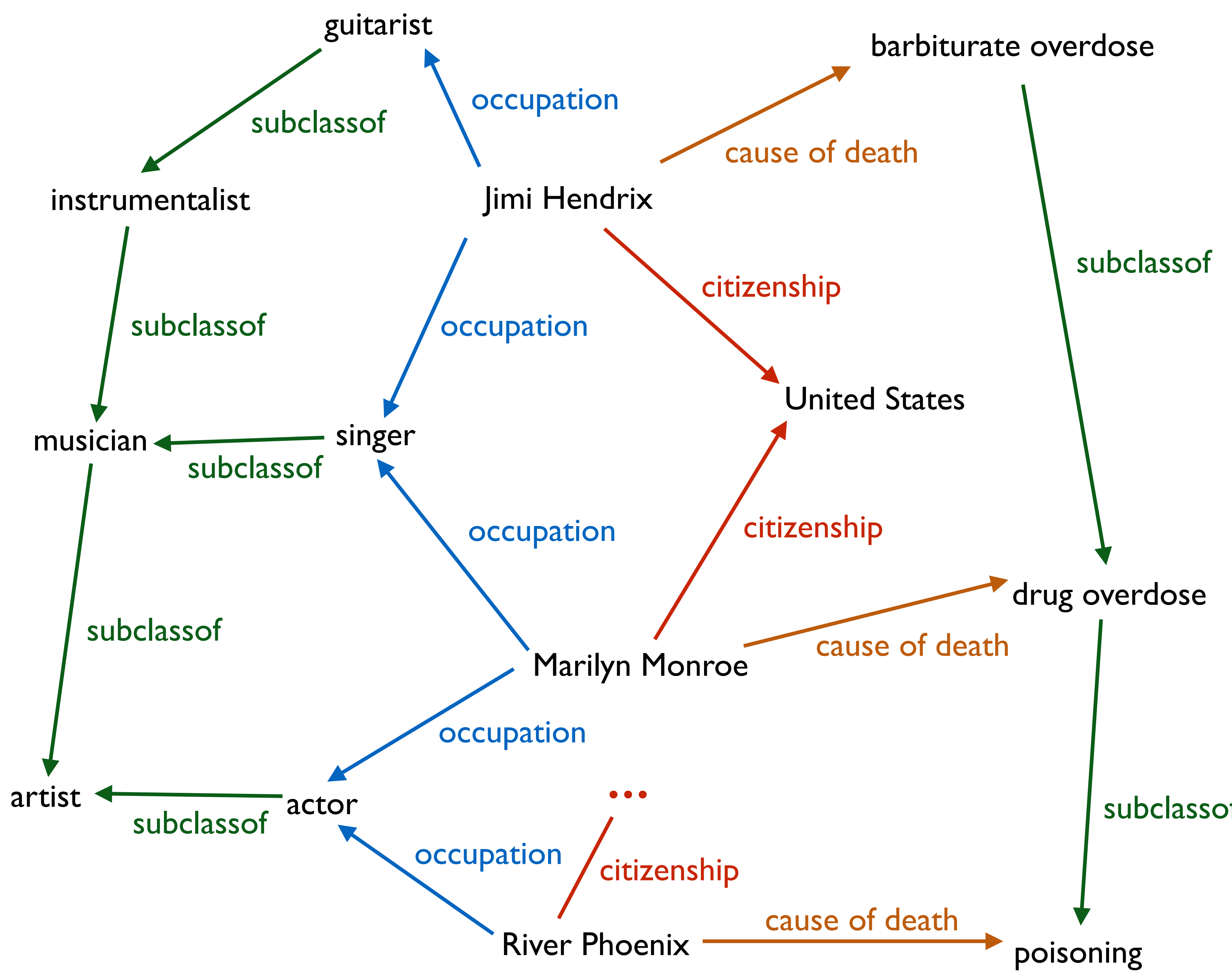
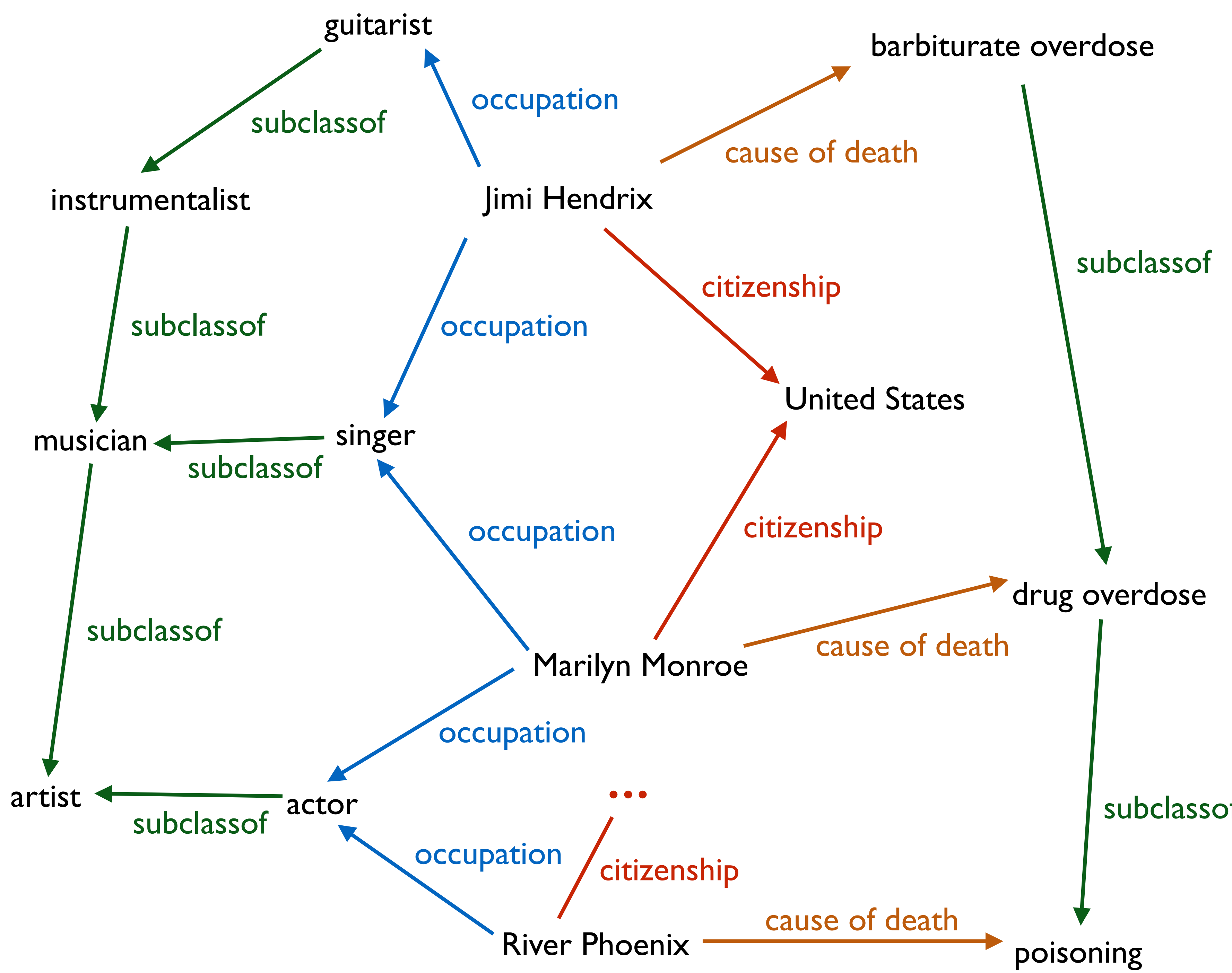
Graph databases

Graph Databases?

Graph Databases?

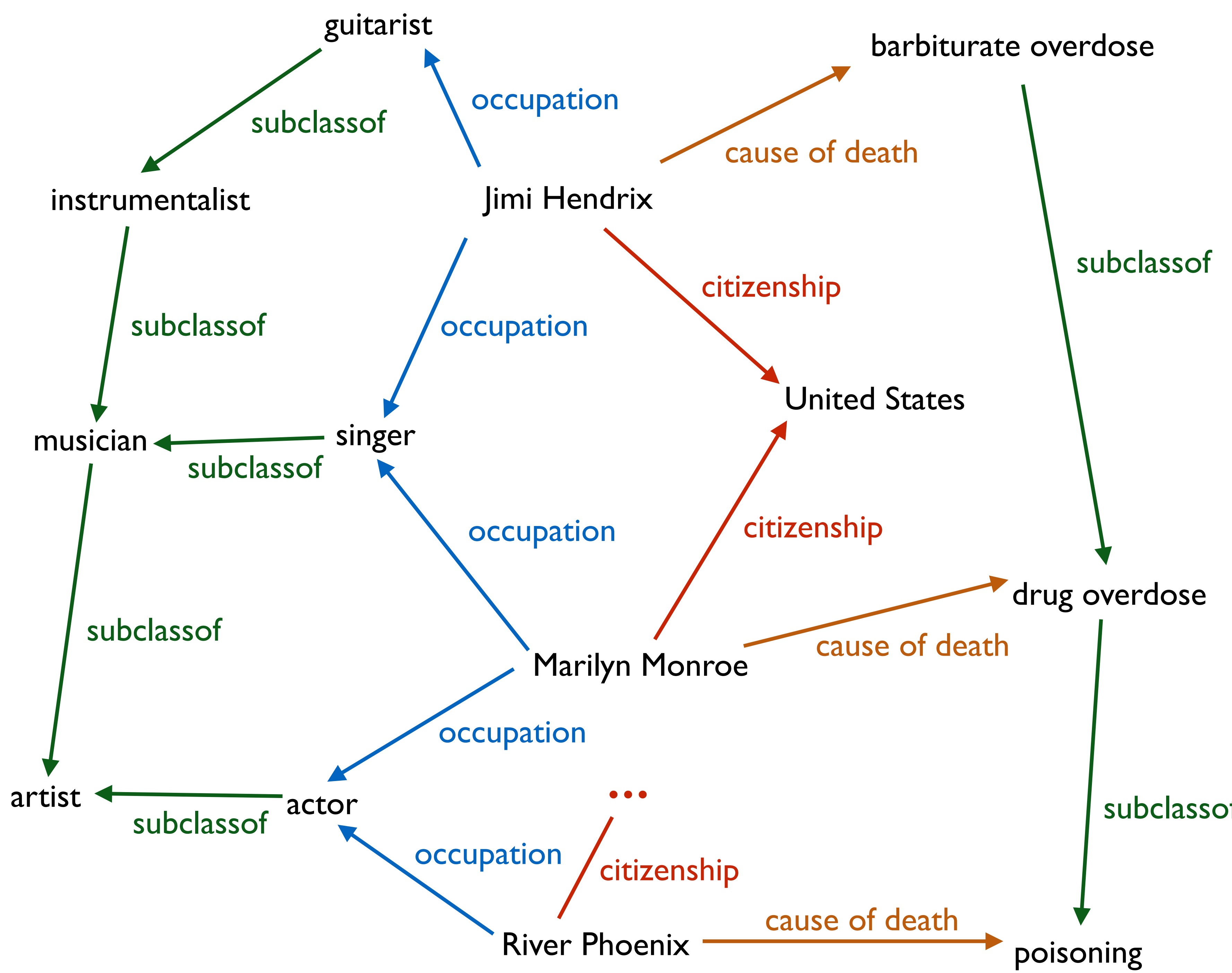


Graph Databases?



Information stored as a graph

Graph Databases?

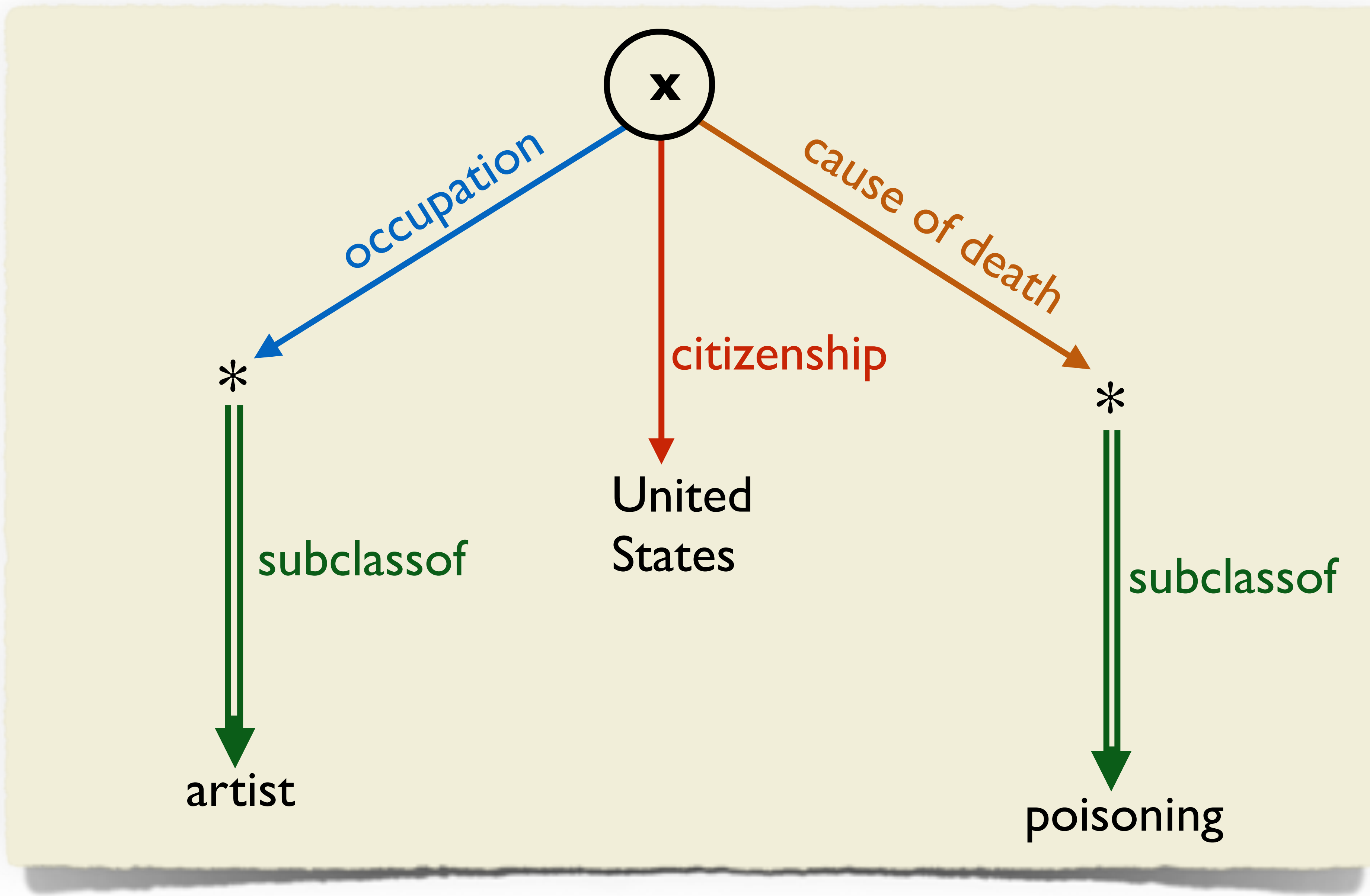


Information stored as a graph

Rather intuitive

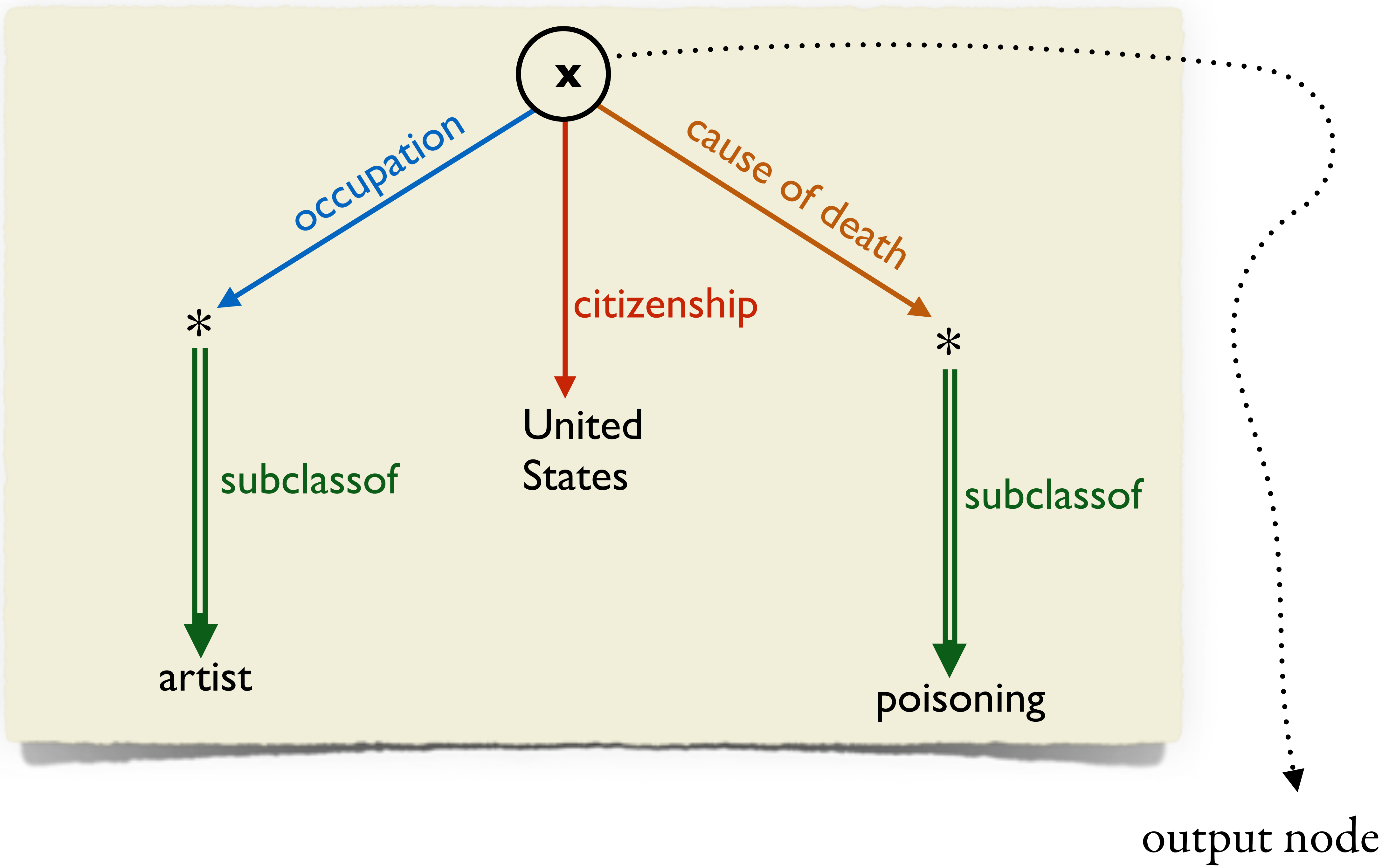
The Query, Visualized

"US artists who died of poisoning"



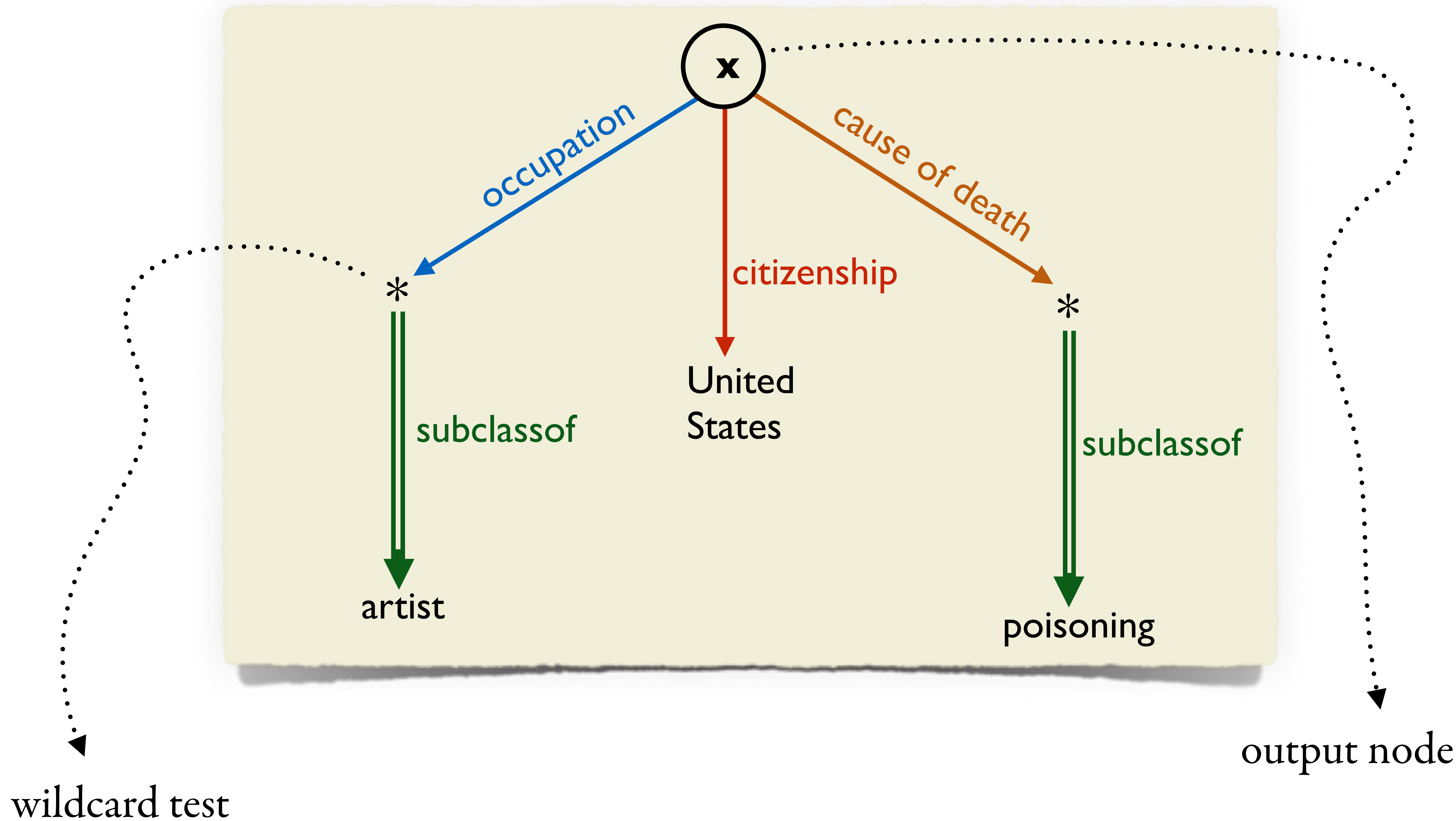
The Query, Visualized

"US artists who died of poisoning"



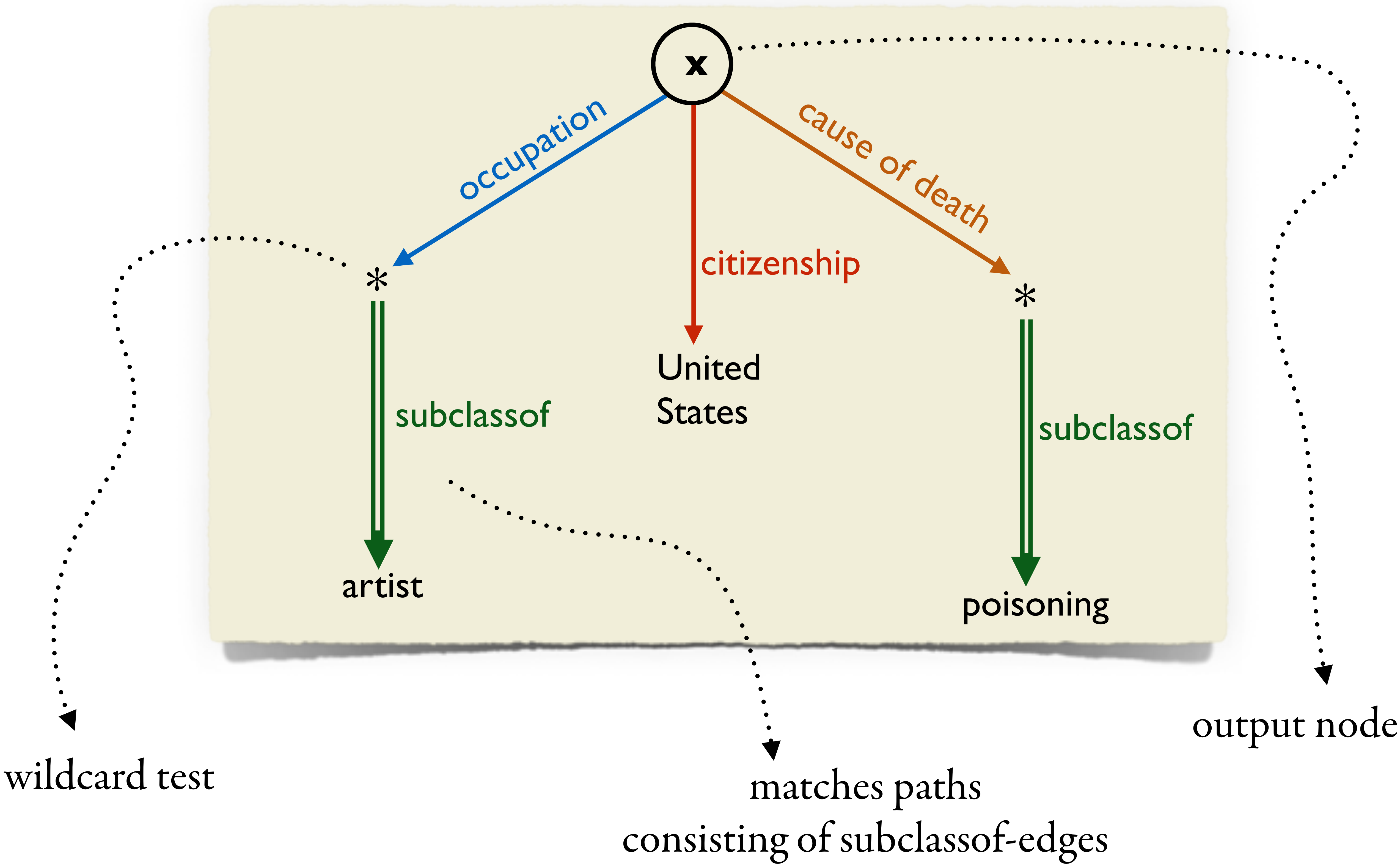
The Query, Visualized

"US artists who died of poisoning"



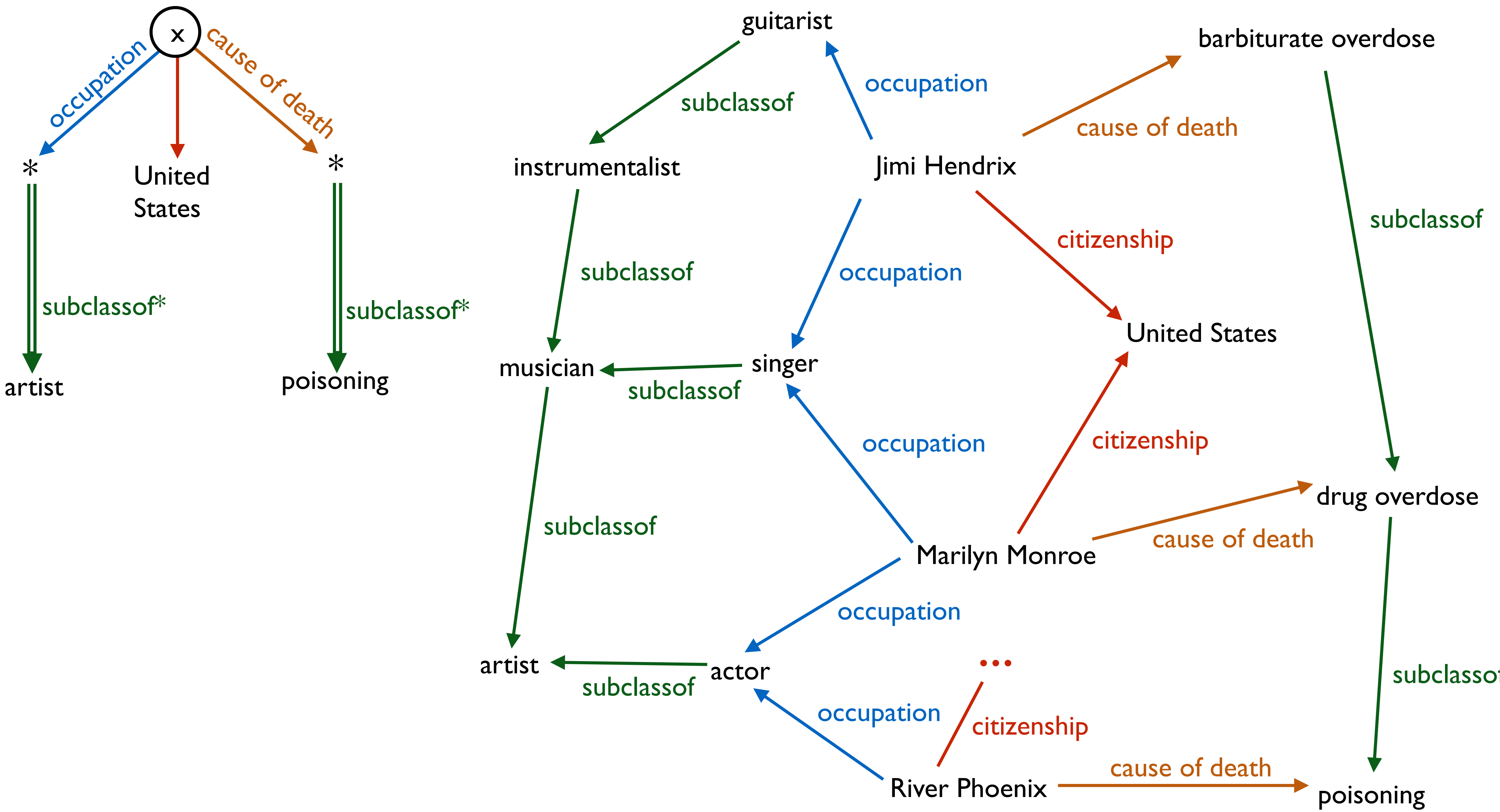
The Query, Visualized

"US artists who died of poisoning"



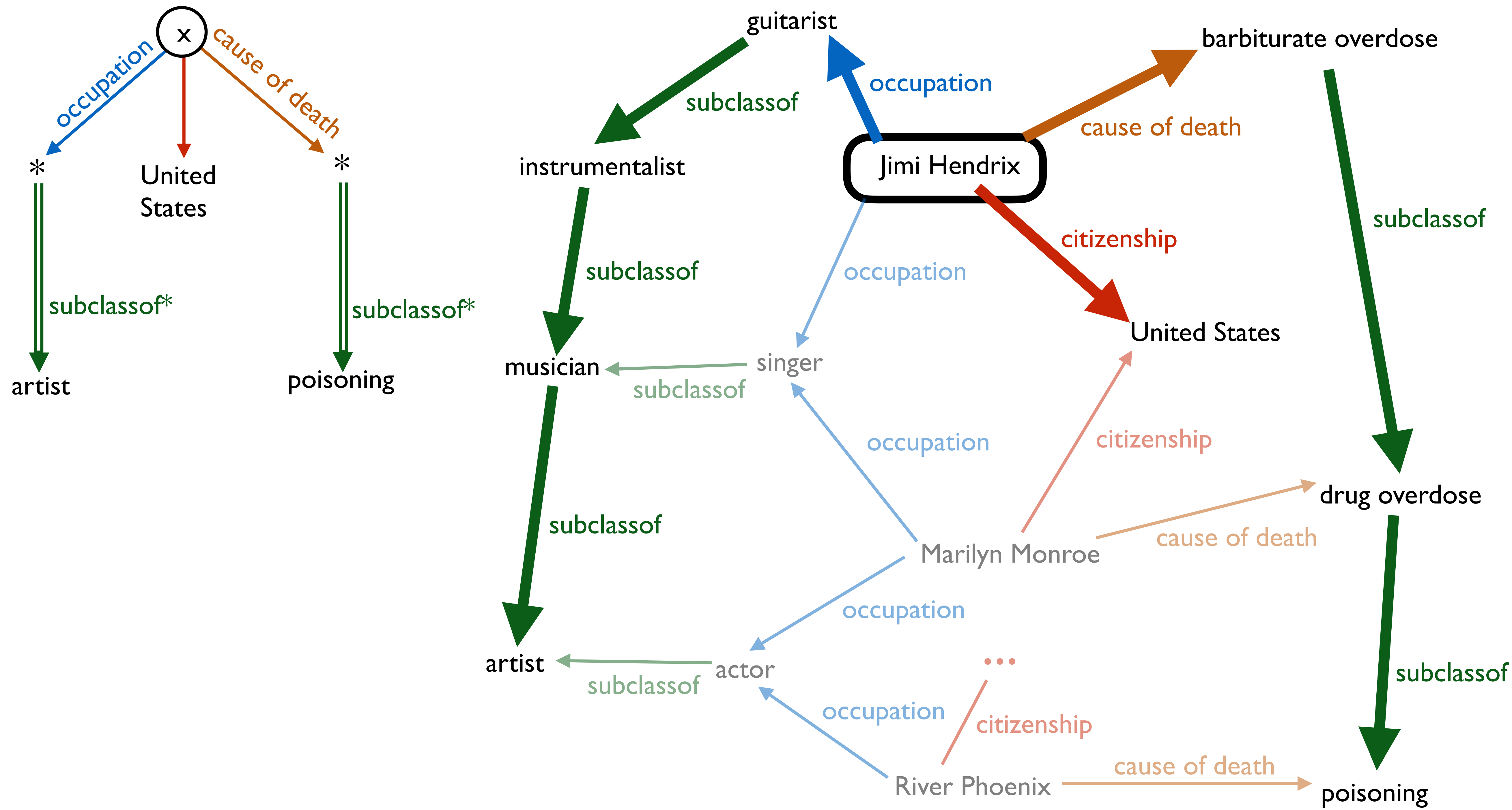
Graph Queries By Example

"US artists who died of poisoning"



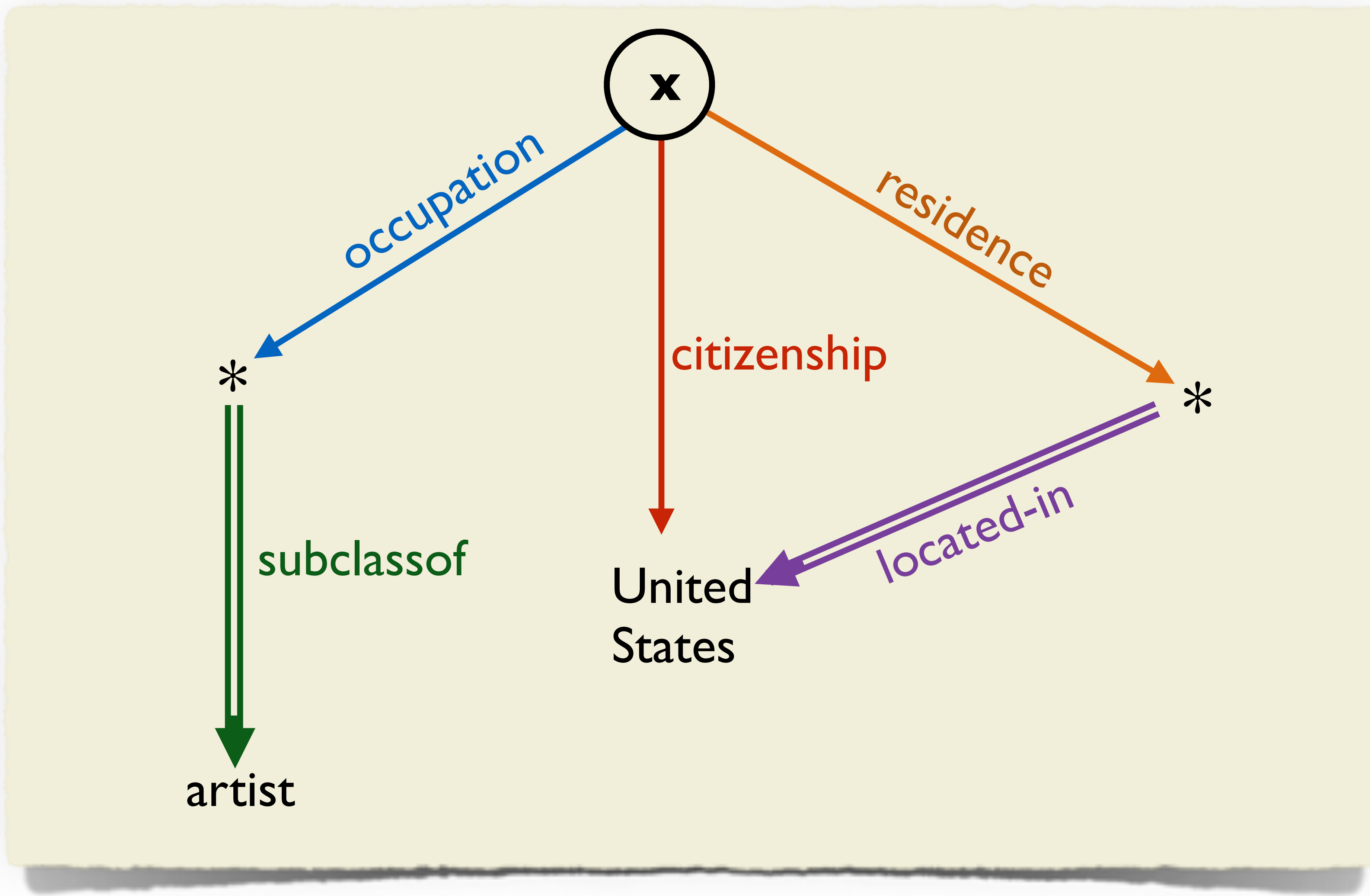
Graph Queries By Example

"US artists who died of poisoning"



Graph Queries By Example

Queries can have cycles



Artists who live in the US and have US citizenship

Why Graph Databases?

Why are they interesting?

Why Graph Databases?

Why are they interesting?

Graph DBs are becoming standard in Industry

Oracle, Neo4j (about 50% of the market), Tigergraph, Redis, SAP, ArangoDB, Amazon Neptune, etc etc
Often hidden: e.g., Google's Knowledge Graph

Why Graph Databases?

Why are they interesting?

Graph DBs are becoming standard in Industry

Oracle, Neo4j (about 50% of the market), Tigergraph, Redis, SAP, ArangoDB, Amazon Neptune, etc etc

Often hidden: e.g., Google's Knowledge Graph

New Standards

ISO is now developing its second database query language standard called GQL: Graph Query Language.

The first one they developed is SQL

Why Graph Databases?

Why are they interesting?

Graph DBs are becoming standard in Industry

Oracle, Neo4j (about 50% of the market), Tigergraph, Redis, SAP, ArangoDB, Amazon Neptune, etc etc
Often hidden: e.g., Google's Knowledge Graph

New Standards

ISO is now developing its second database query language standard called GQL: Graph Query Language.
The first one they developed is SQL

New Applications

Social networks, Semantic Web, bioinformatics, fraud analysis, real-time recommendation, network/IT systems, even investigative journalism (Panama+Pandora papers)

Why Graph Databases?

Why are they interesting?

Why Graph Databases?

Why are they interesting?

Future in Analytics

Gartner prediction: in the next 5 years, up to 80% of all analytics task will involve graph databases

Why Graph Databases?

Why are they interesting?

Future in Analytics

Gartner prediction: in the next 5 years, up to 80% of all analytics task will involve graph databases

Growth potential

IDG prediction: 600% growth up to 2025

Why Graph Databases?

Why are they interesting?

Future in Analytics

Gartner prediction: in the next 5 years, up to 80% of all analytics task will involve graph databases

Growth potential

IDG prediction: 600% growth up to 2025

Current and future use

75% of Fortune 100 companies currently use graph databases

Phenomenal fundraising (last year alone, around 500M)

GQL Influence Graph

GQL Influence Graph

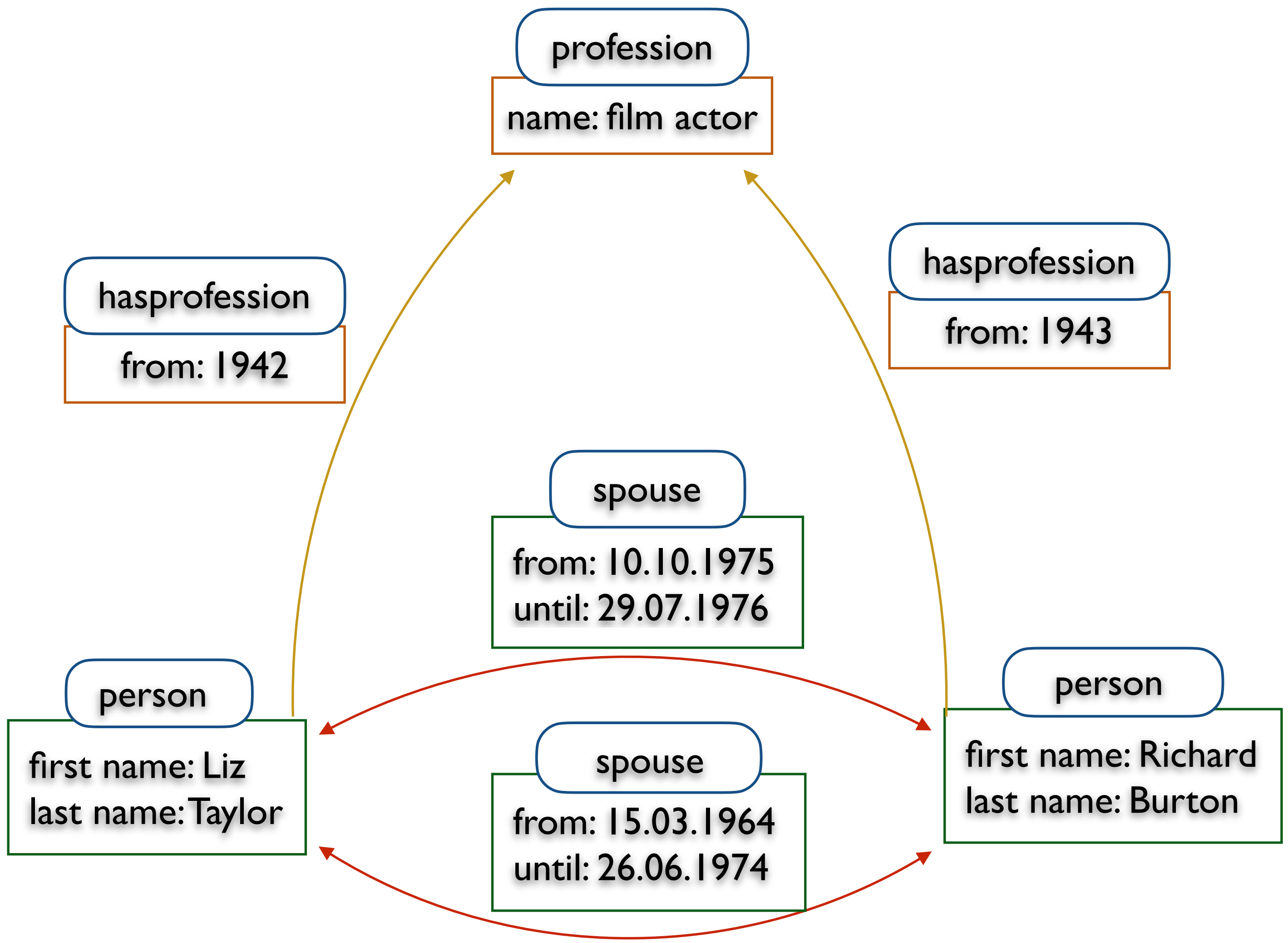
[https://www.gqlstandards.org/existing-languages]

Models for Graph Databases?

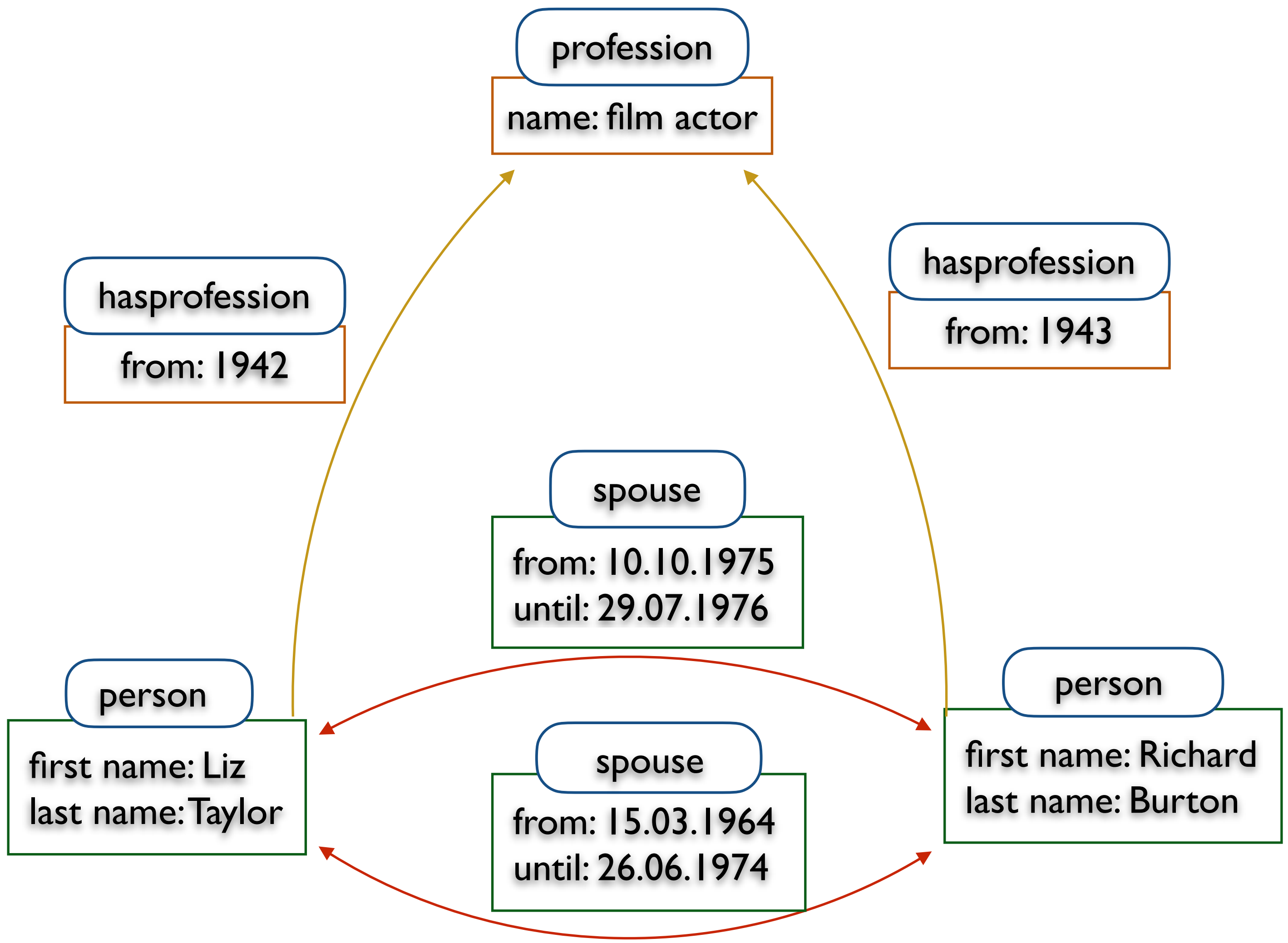
Currently, two main data models:

- Property Graph Databases (today: the dominant model)
- RDF-like Databases (an earlier and interesting approach but not as prevalent in industry)

Property Graph Data Model

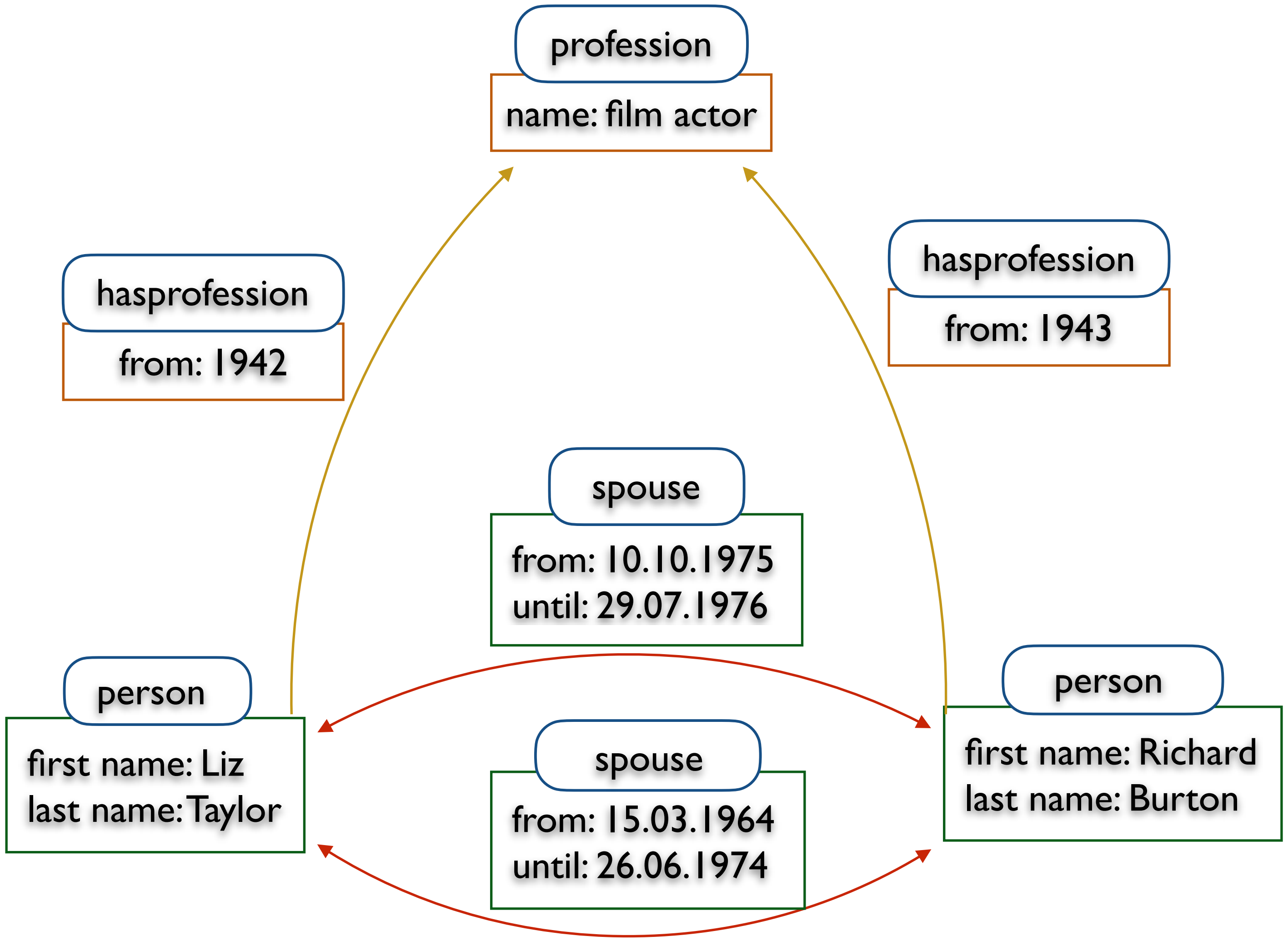


Property Graph Data Model



More formally, this is

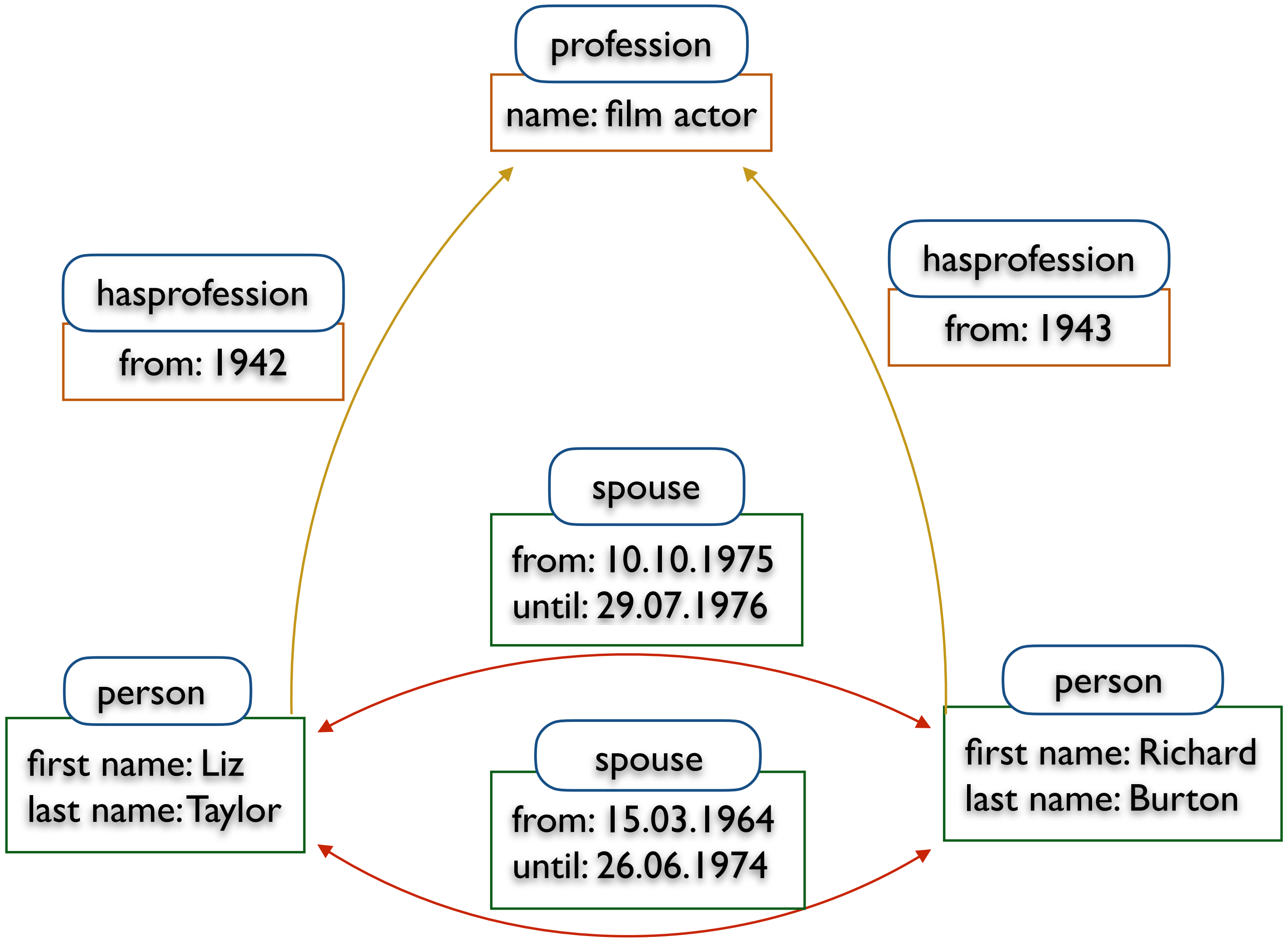
Property Graph Data Model



More formally, this is

- a set of node identifiers N

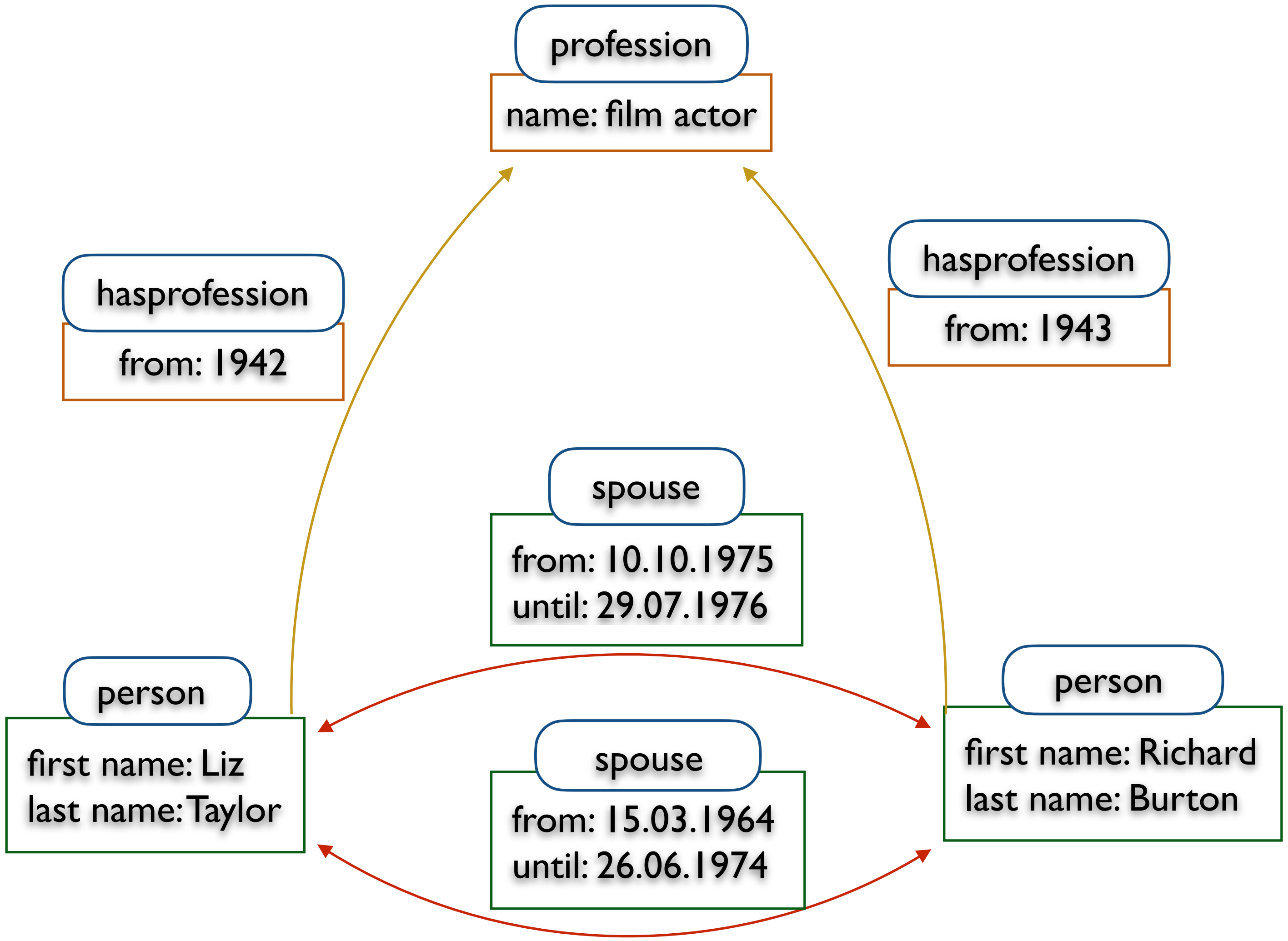
Property Graph Data Model



More formally, this is

- a set of node identifiers N
- a set of edge identifiers E

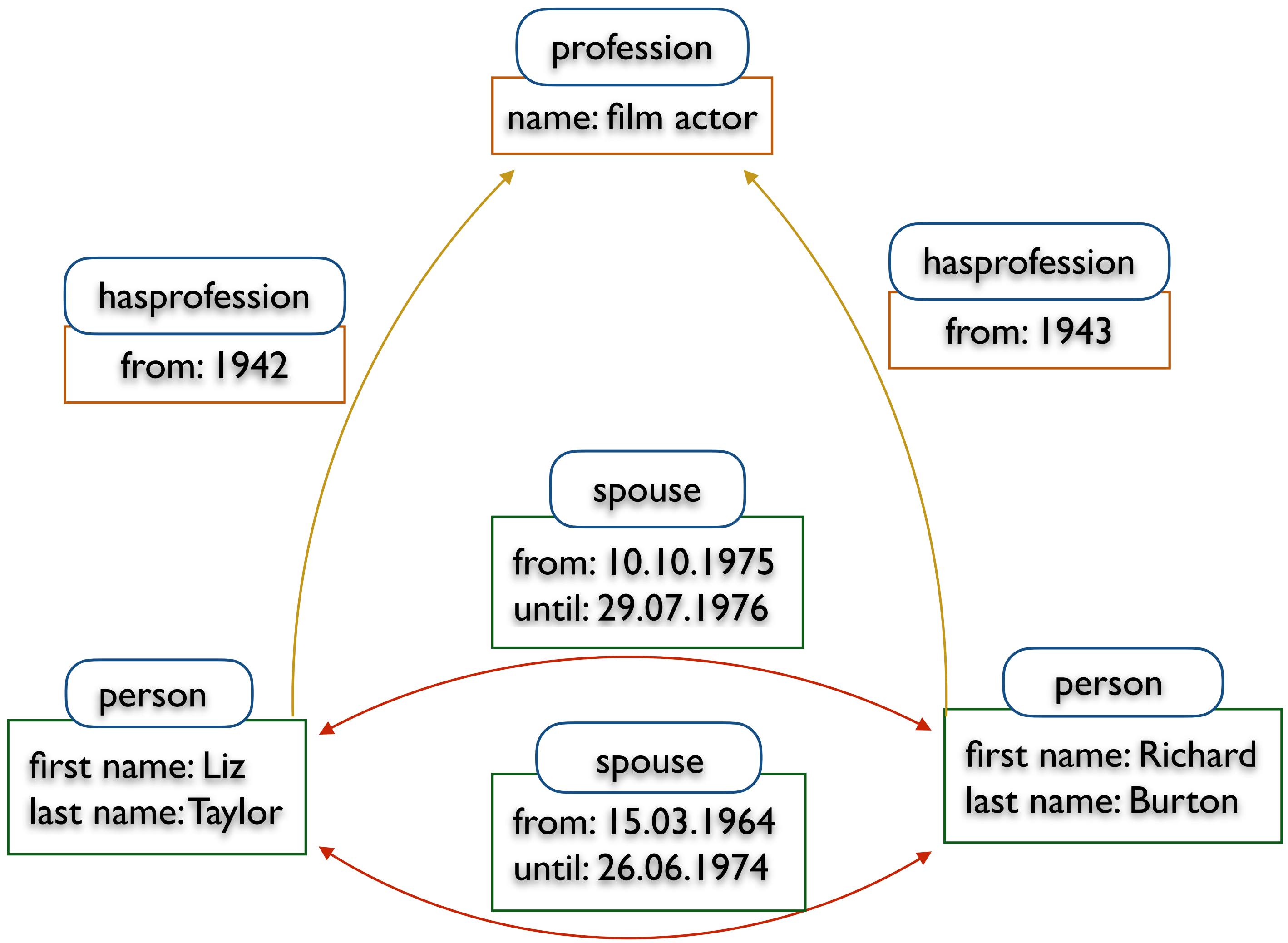
Property Graph Data Model



More formally, this is

- a set of node identifiers N
- a set of edge identifiers E
- a function that maps E to N × N

Property Graph Data Model

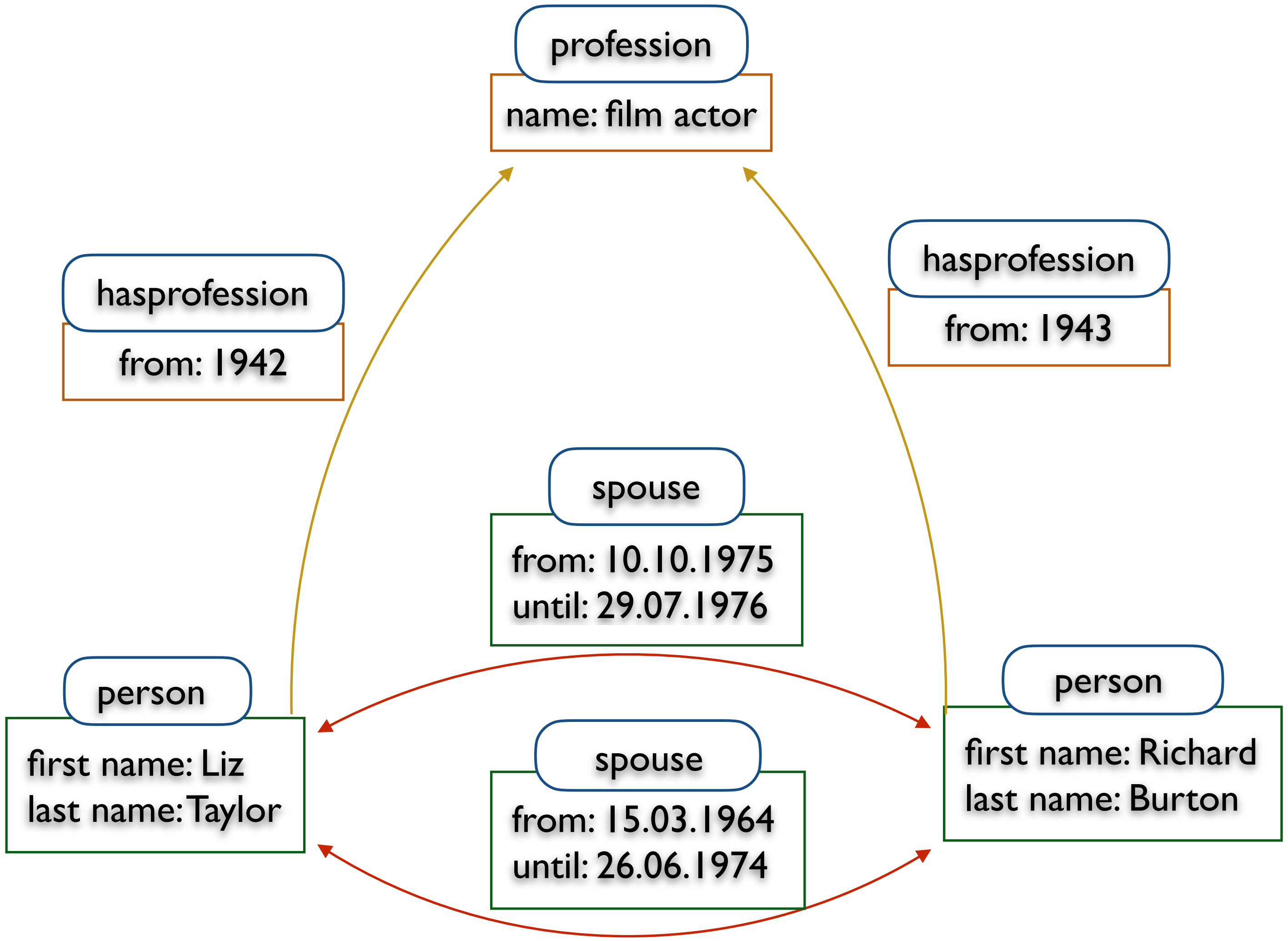


Labels L: person, profession, spouse

More formally, this is

- a set of node identifiers N
- a set of edge identifiers E
- a function that maps E to $N \times N$

Property Graph Data Model



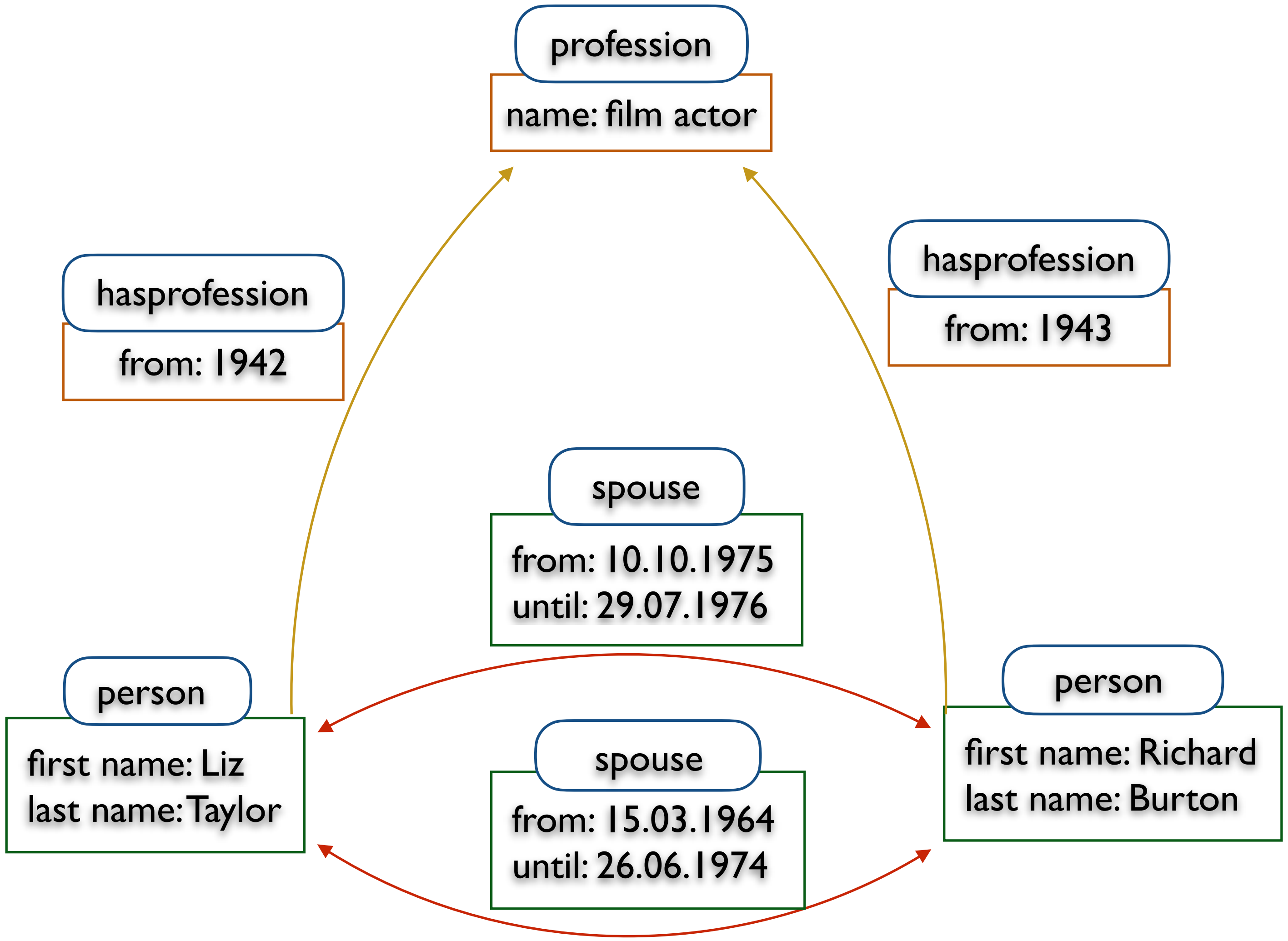
Labels **L**: person, profession, spouse

Values **V**: Liz, Taylor, 10.10.1975

More formally, this is

- a set of node identifiers N
- a set of edge identifiers E
- a function that maps E to $N \times N$

Property Graph Data Model



Labels **L**: person, profession, spouse

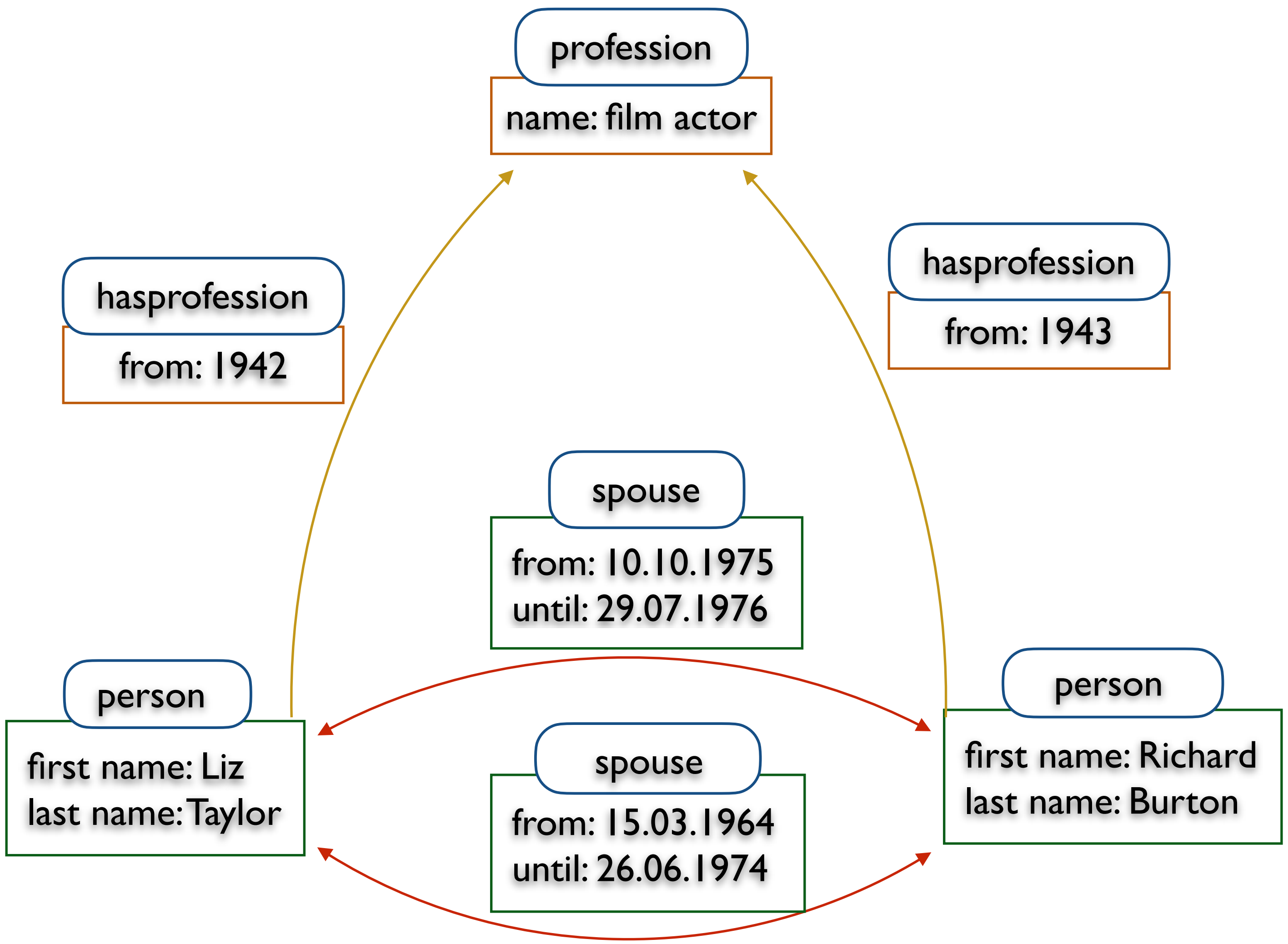
Values **V**: Liz, Taylor, 10.10.1975

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 $N \times N$

Property Graph Data Model



Labels **L**: person, profession, spouse

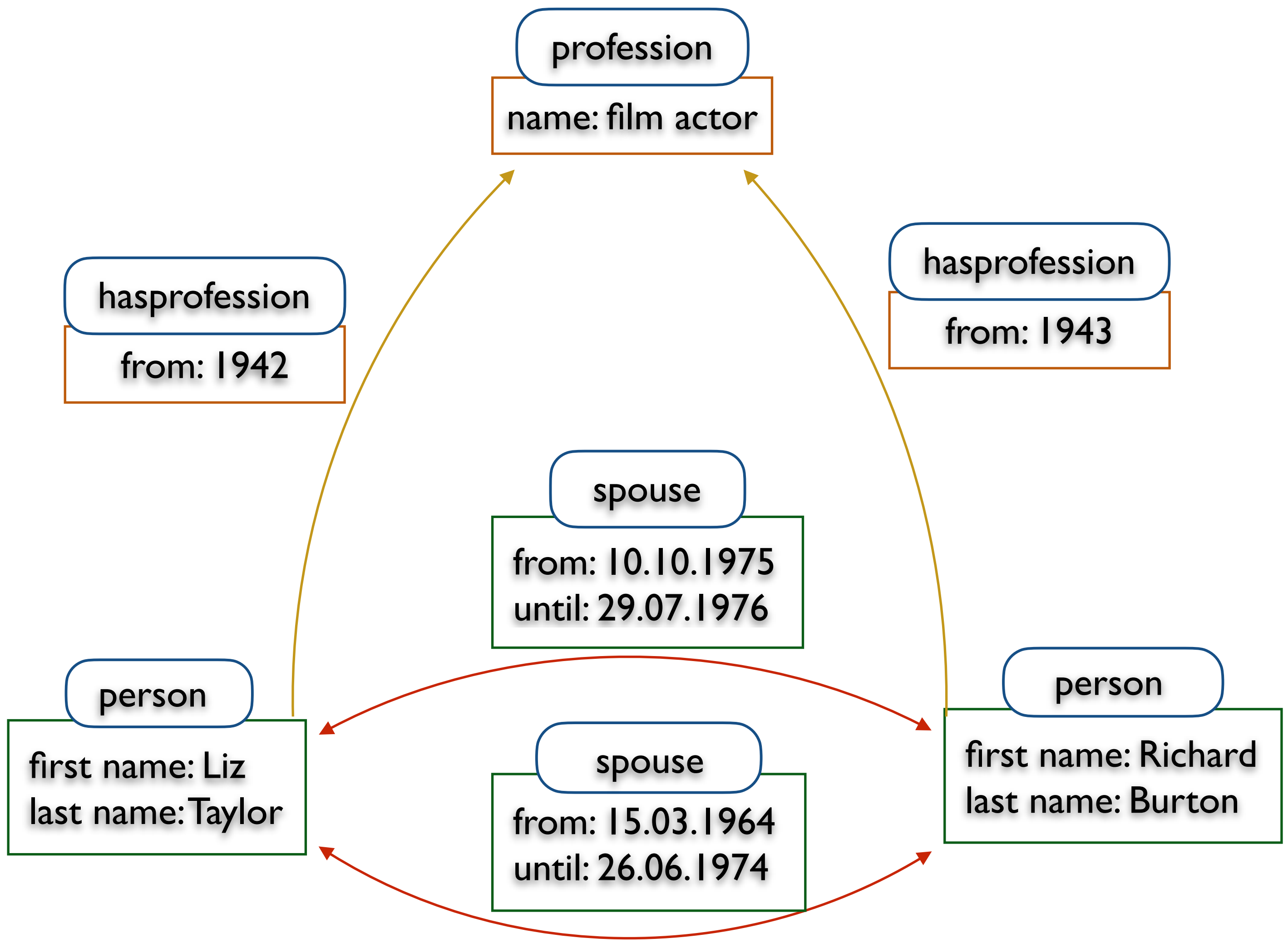
Values **V**: Liz, Taylor, 10.10.1975

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 × N
- a function from N ∪ E to (subsets of) labels **L**

Property Graph Data Model



Labels **L**: person, profession, spouse

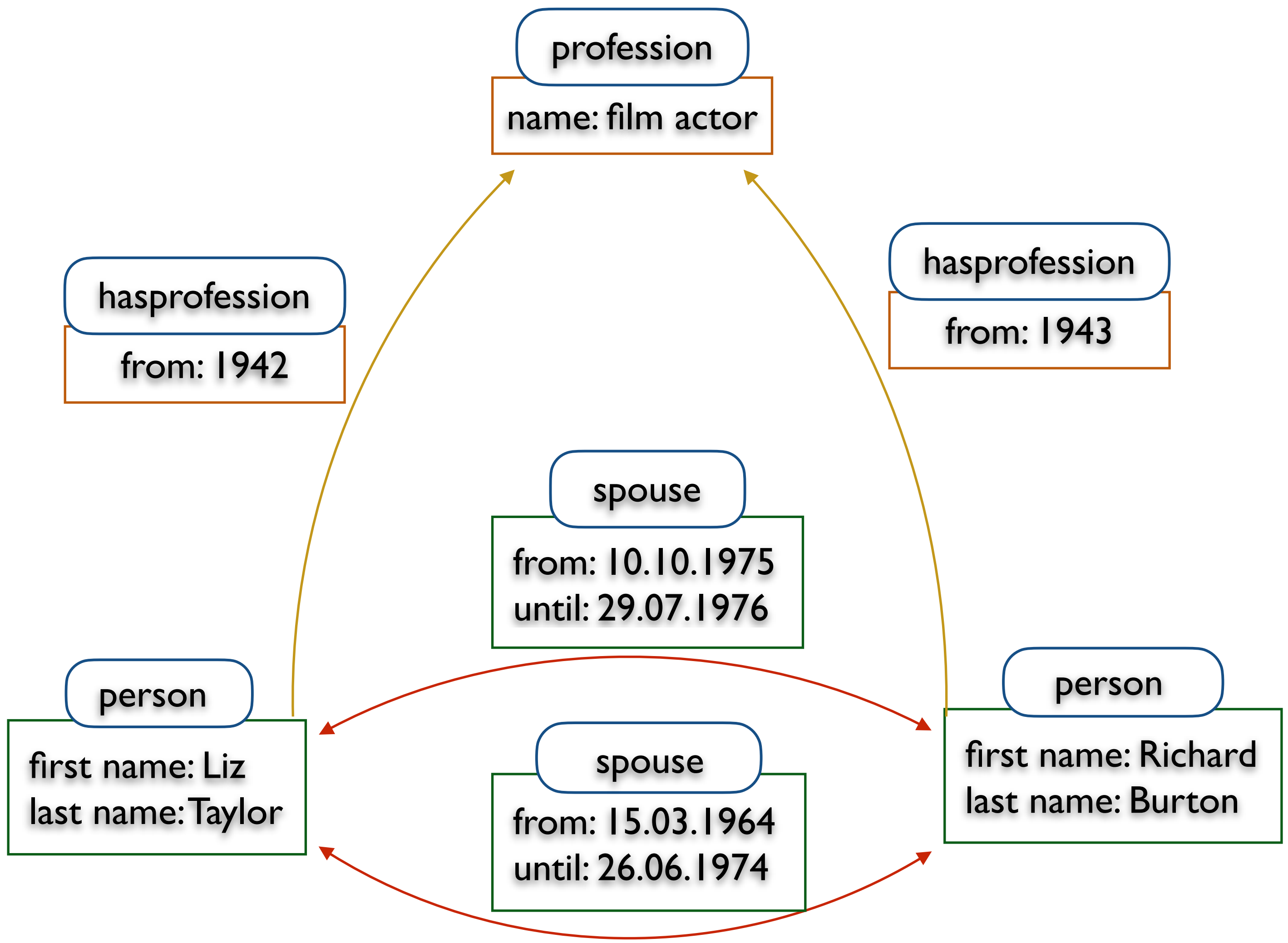
Values **V**: Liz, Taylor, 10.10.1975

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**

Property Graph Data Model



Labels **L**: person, profession, spouse

Values **V**: Liz, Taylor, 10.10.1975

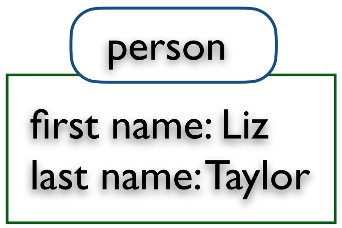
Properties **P**: first name, last name

More formally, this is

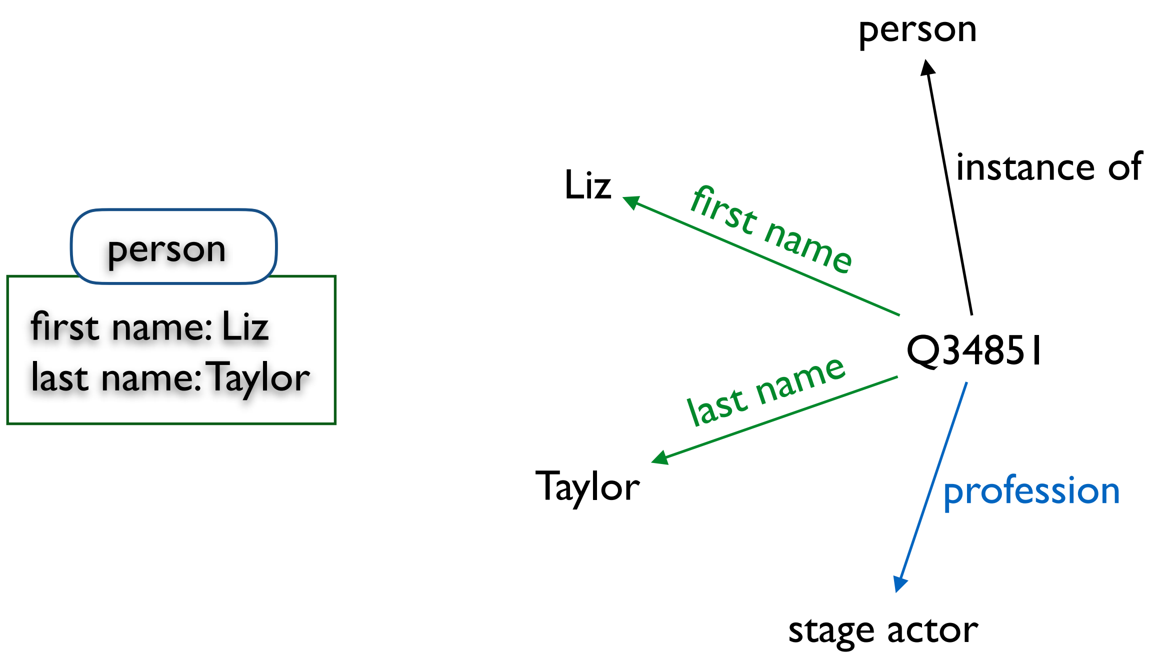
- a set of node identifiers N
- a set of edge identifiers E
- a function that maps E to $N \times N$
- a function from $N \cup E$ to (subsets of) labels **L**
- a function from $(N \cup E) \times \mathbf{P}$ to (subsets of) values **V**

Some models also directly incorporate paths

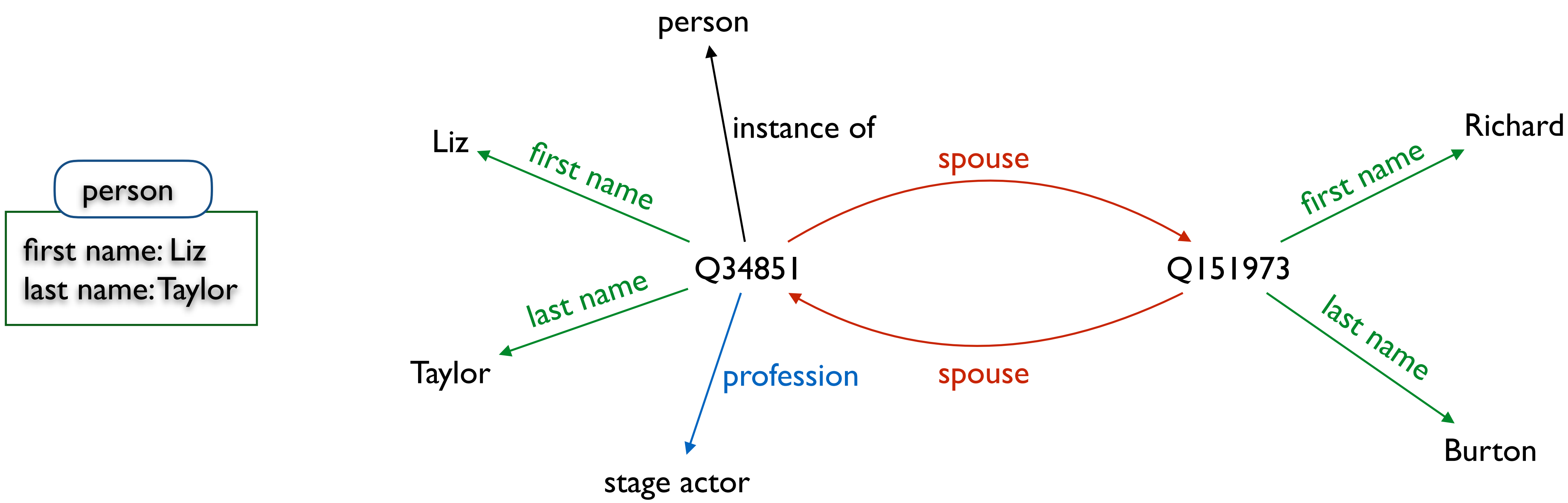
RDF Data Model



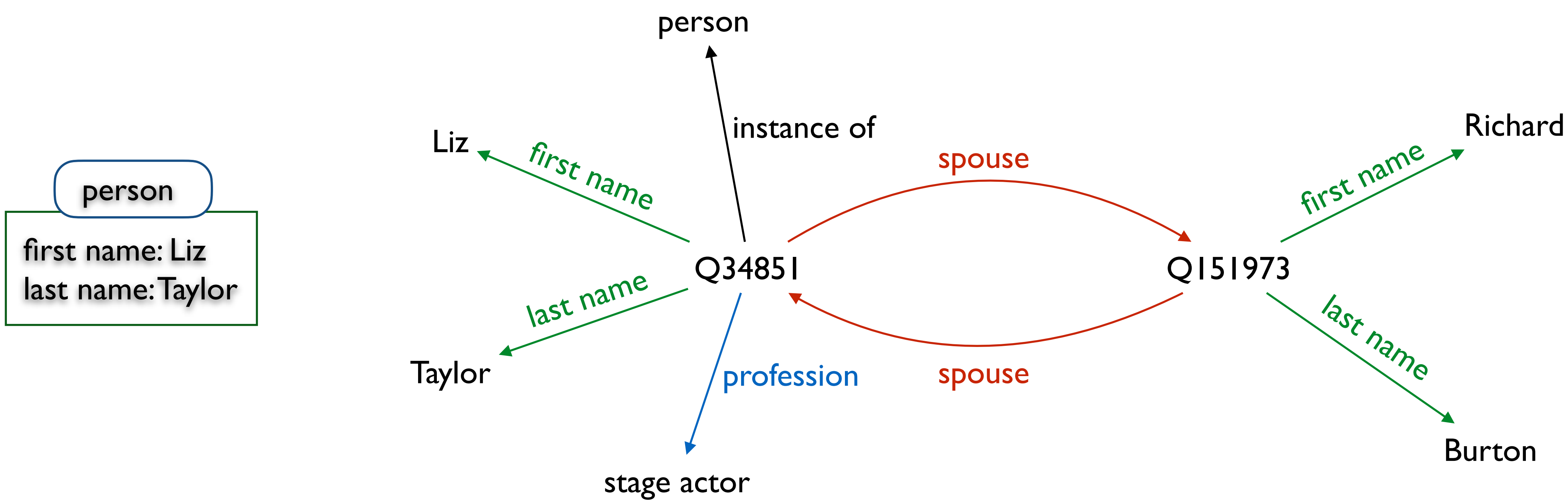
RDF Data Model



RDF Data Model

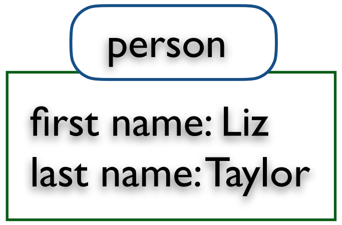


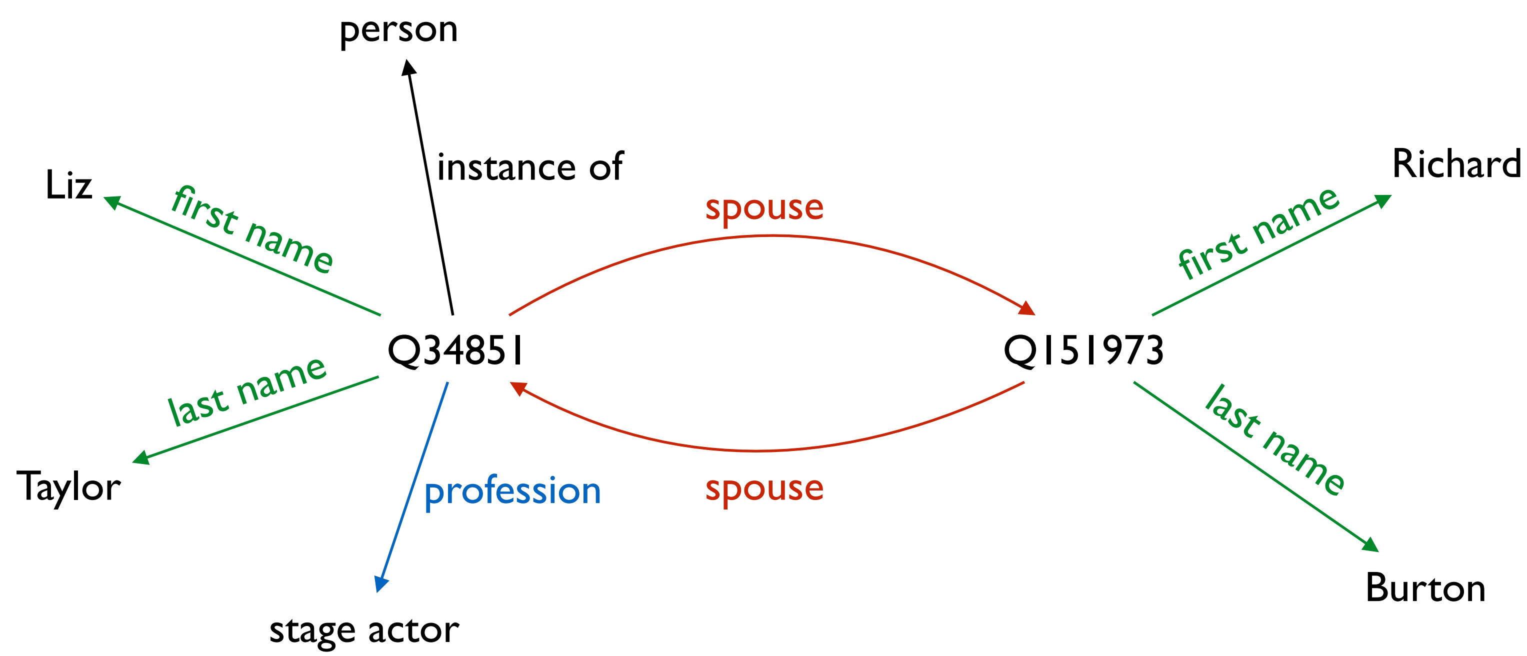
RDF Data Model



More formally, this is a set of triples from

RDF Data Model

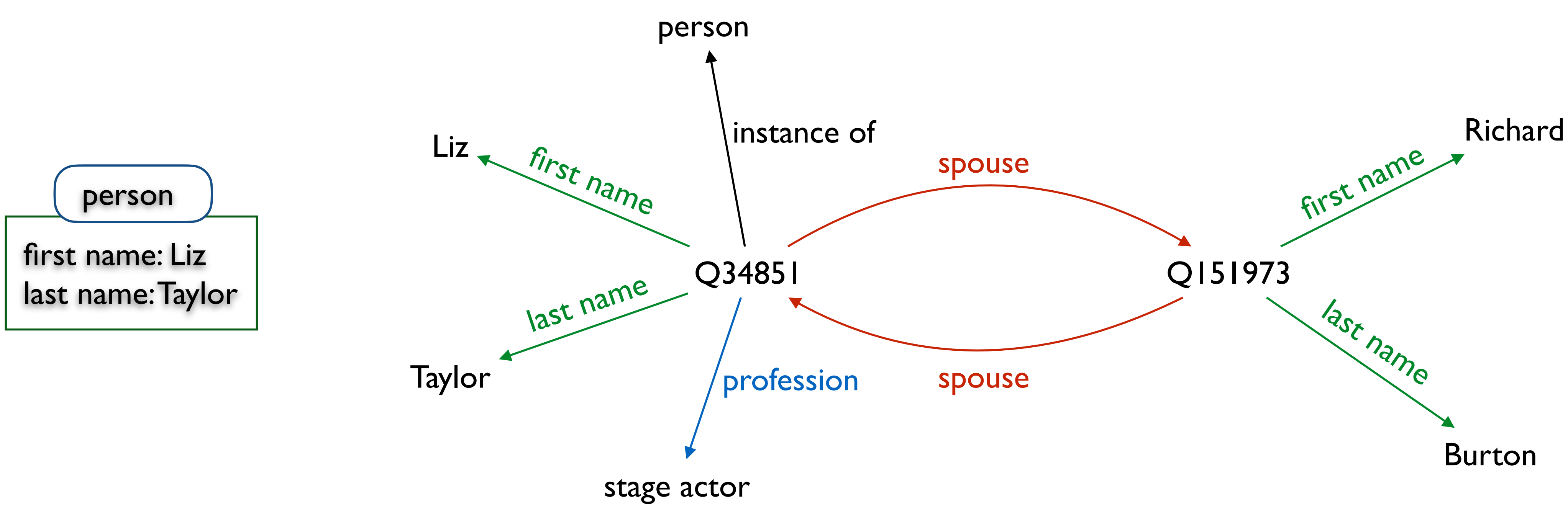




More formally, this is a set of triples from

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

RDF Data Model

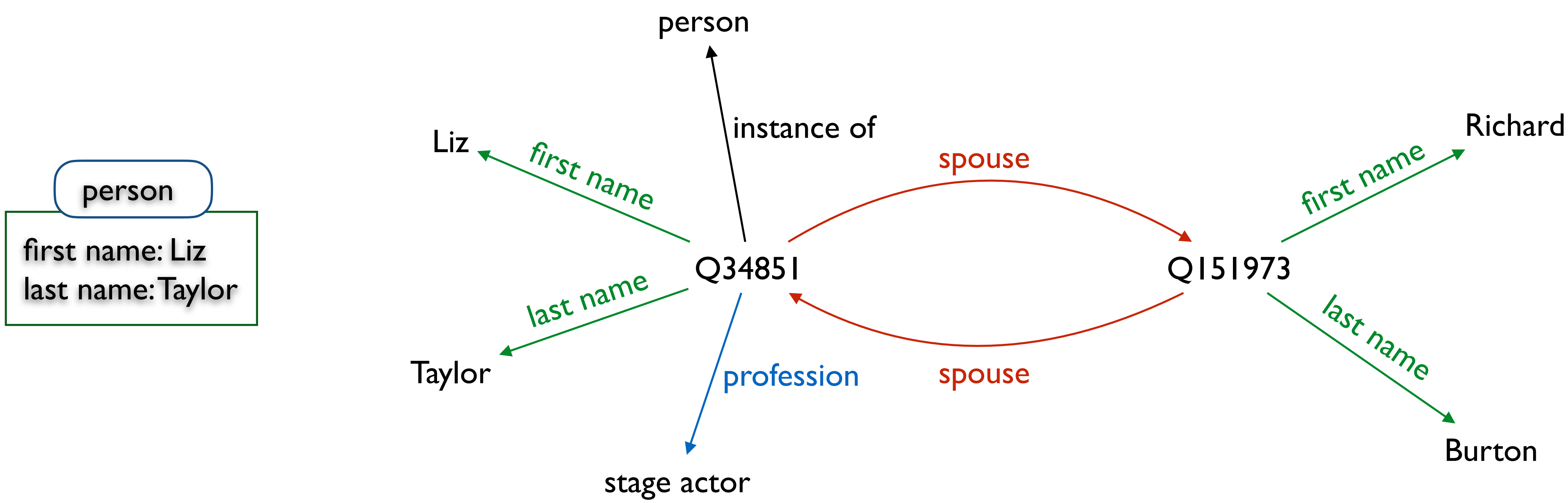


More formally, this is a set of triples from

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

where

RDF Data Model



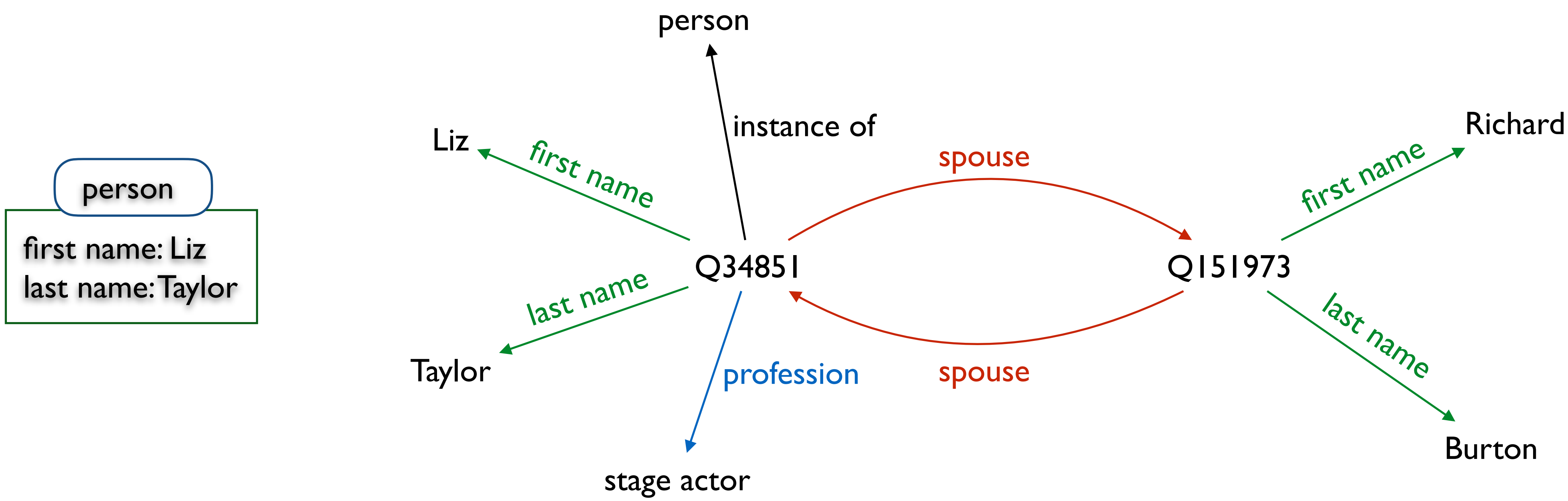
More formally, this is a set of triples from

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

where

- I is the set of Internationalized Resource Identifiers (IRIs)

RDF Data Model



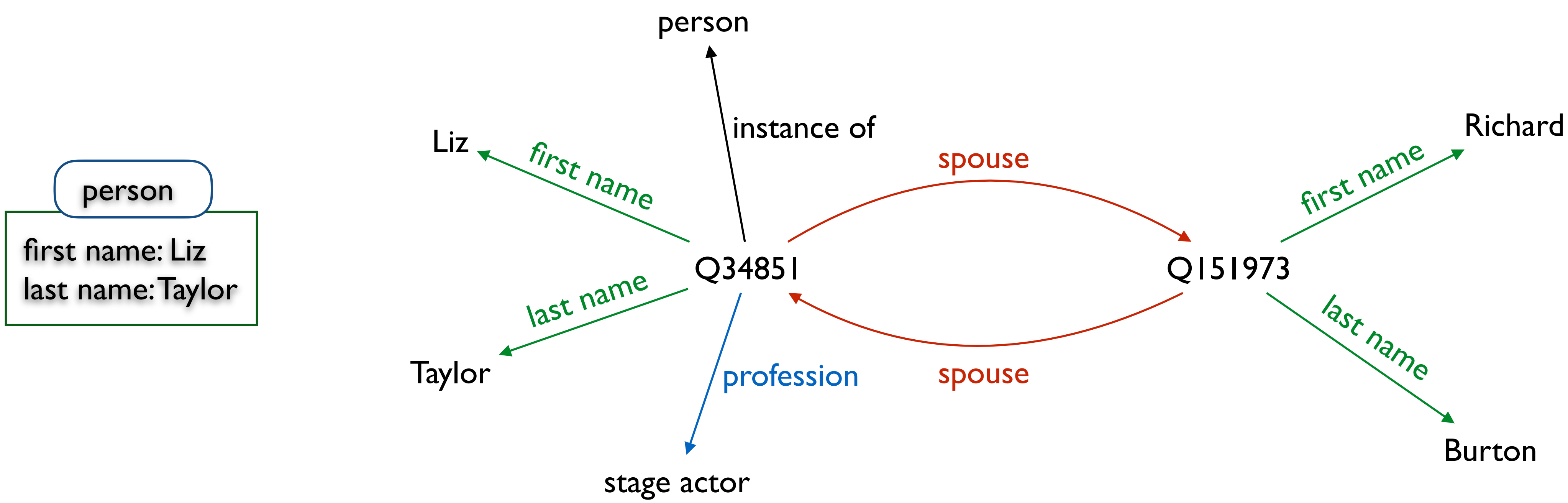
More formally, this is a set of triples from

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

where

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

RDF Data Model



More formally, this is a set of triples from

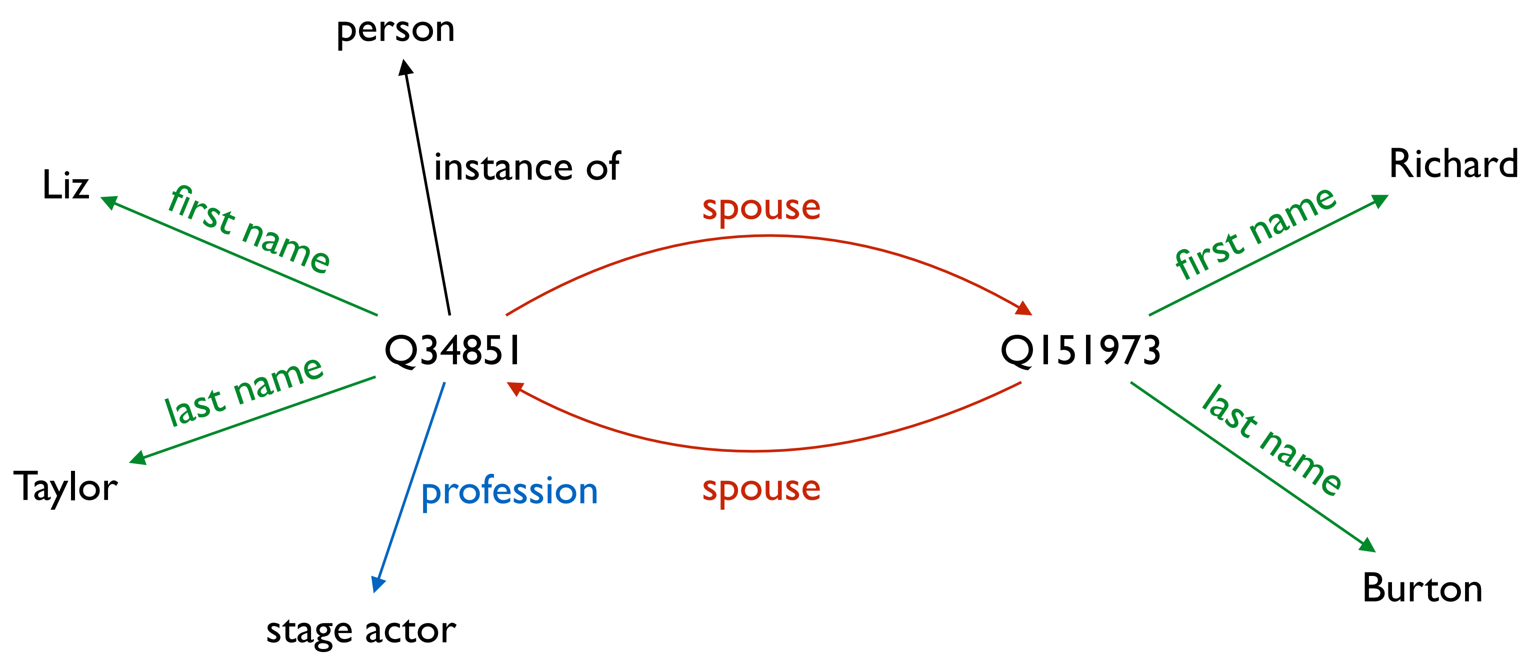
$$I \times I \times (I \cup L)$$

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

Most theoretical development is based on



Edge-labeled, directed graphs

Graph Database

We assume that Σ is a countably infinite set of labels

Graph Database

We assume that Σ is a countably infinite set of labels

Definition

A **graph database** (over Σ) 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

Building blocks of query languages: RPQs and CRPQs

Building blocks of query languages: RPQs and CRPQs

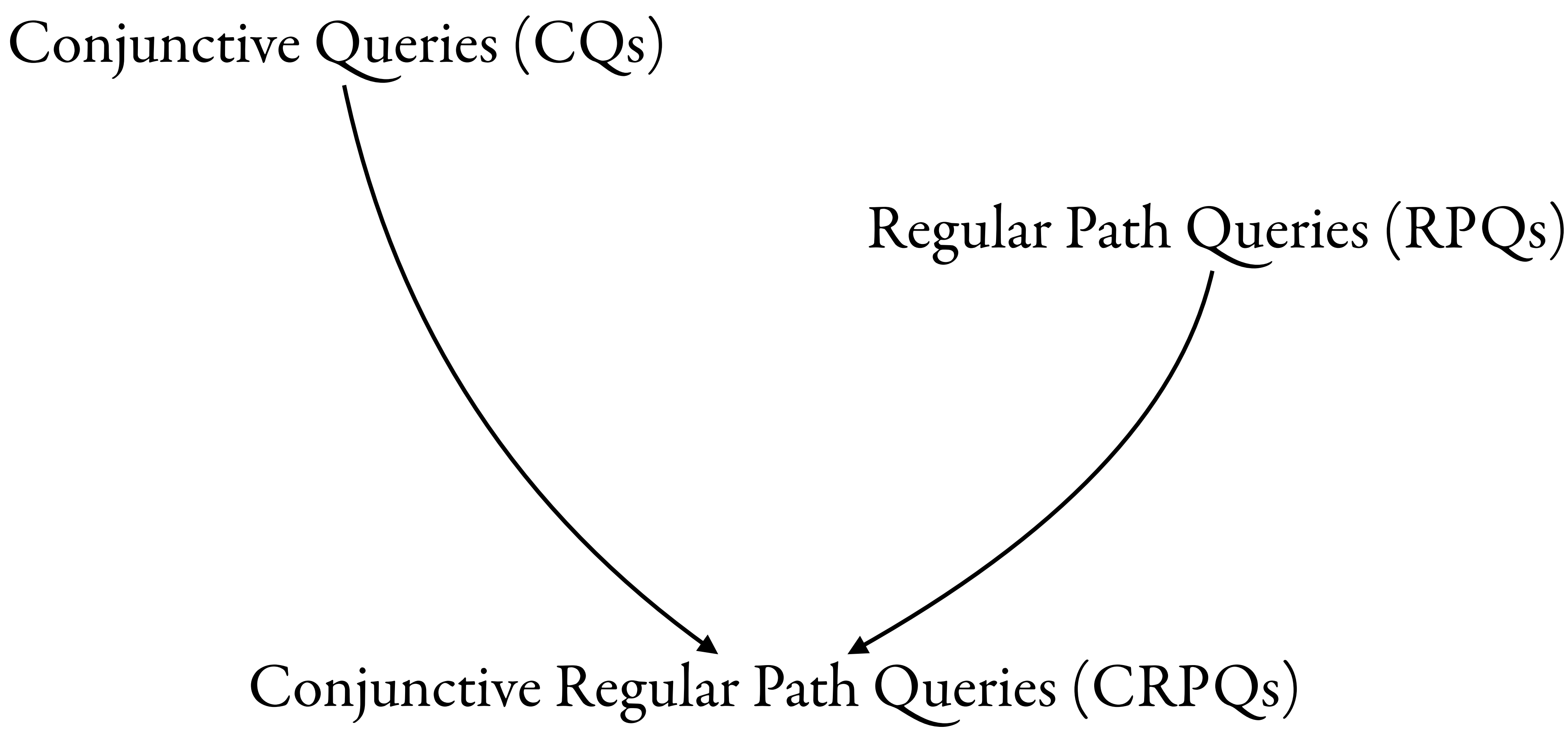
Conjunctive Queries (CQs)

Building blocks of query languages: RPQs and CRPQs

Conjunctive Queries (CQs)

Regular Path Queries (RPQs)

Building blocks of query languages: RPQs and CRPQs



Notation and Basic Principles

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

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

(So we write a^4 for $aaaa$)

Regular Path Queries

Why regular path queries?

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

they can only express local properties

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

Definition

A path in graph G is a sequence

$$p = (v_0, a_1, v_1)\,(v_1, a_2, v_2)\,\ldots\,(v_{n\text{-}2}, a_n, v_{n\text{-}1})\,(v_{n\text{-}1}, a_n, v_n)$$

of edges of G. Label of p is $a_1 a_2 \cdots a_n$

Regular Path Queries

Definition

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

$$x \xrightarrow{r} y$$

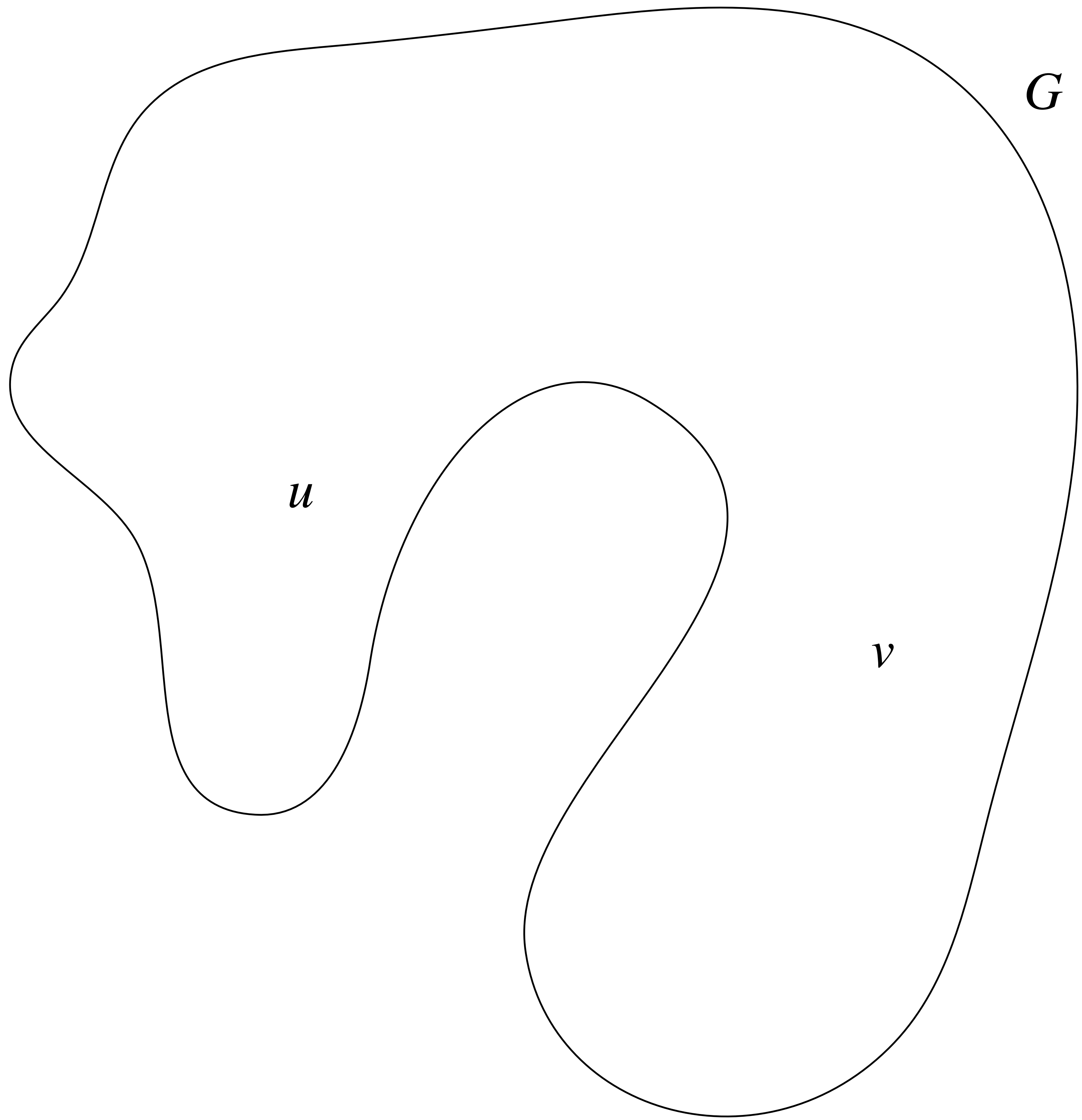
where x and y are variables and r is a regular expression over Σ

(Notice that r can only mention a finite subset of Σ)

Semantics of RPQs

RPQ

$x \xrightarrow{r} y$



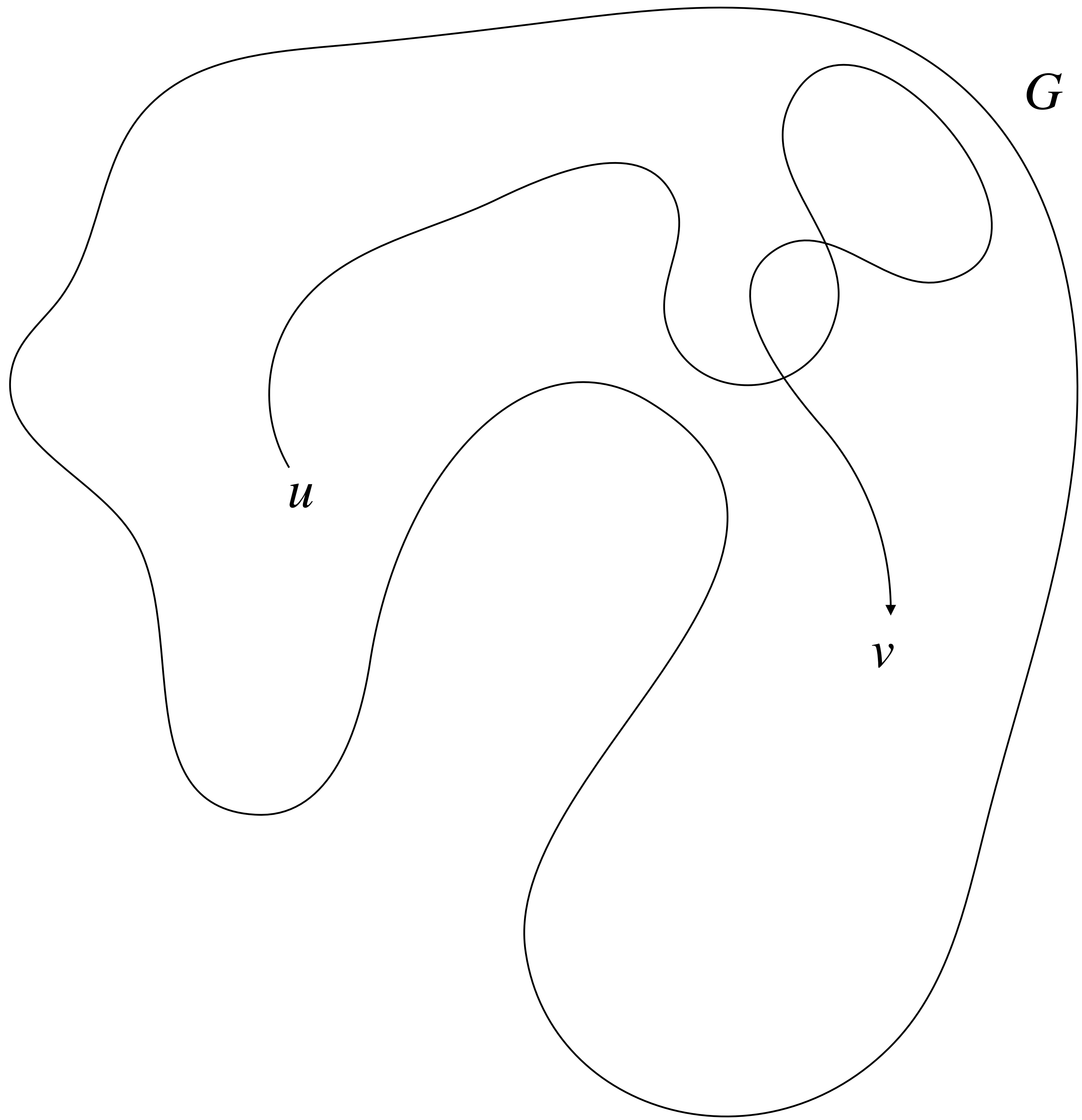
(u,v) is returned iff
there is a path
from u to v
whose label matches r

Semantics of RPQs

RPQ

$x \xrightarrow{r} y$

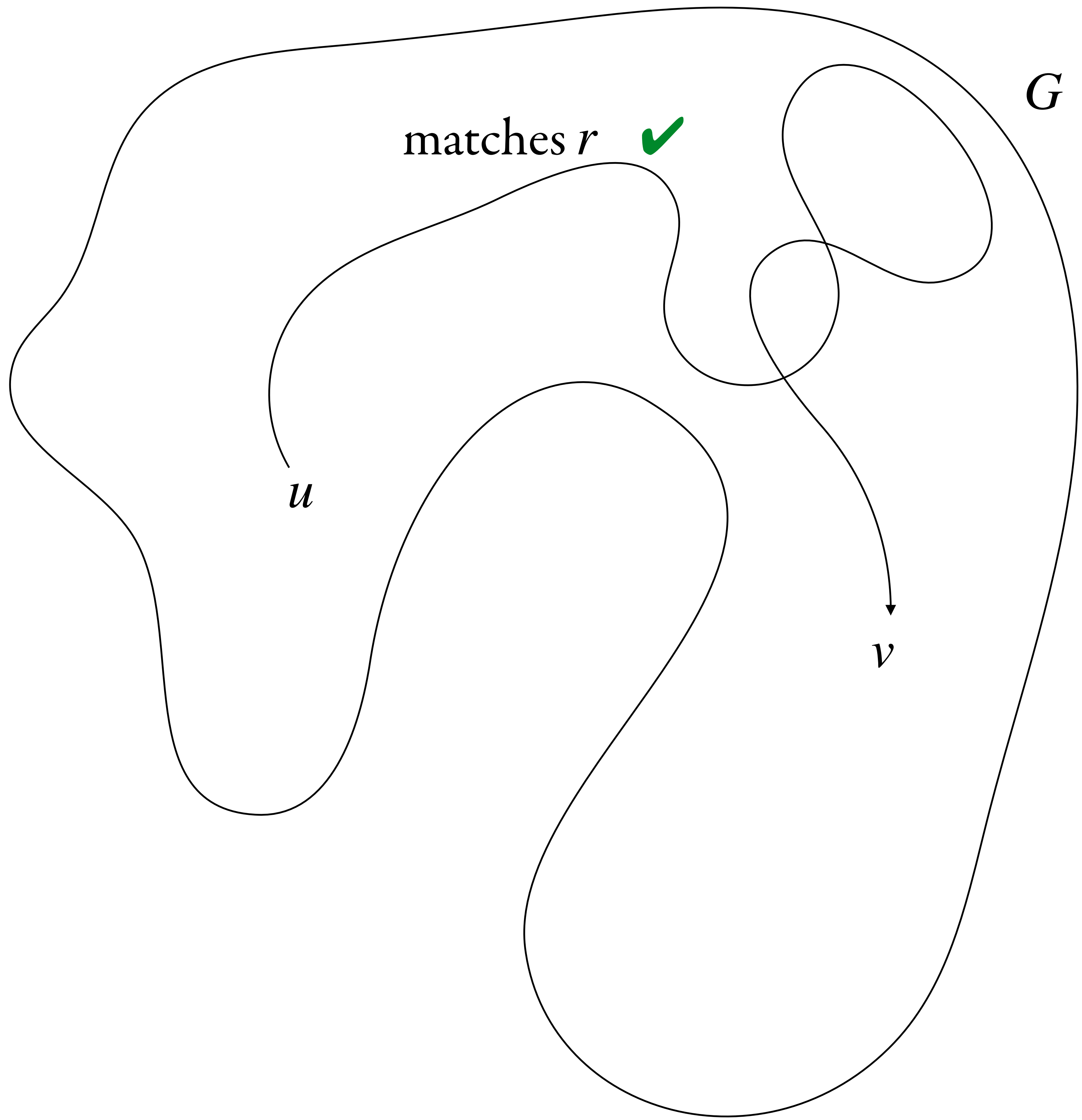
(*u*,*v*) is returned iff
there is a path
from *u* to *v*
whose label matches *r*



Semantics of RPQs

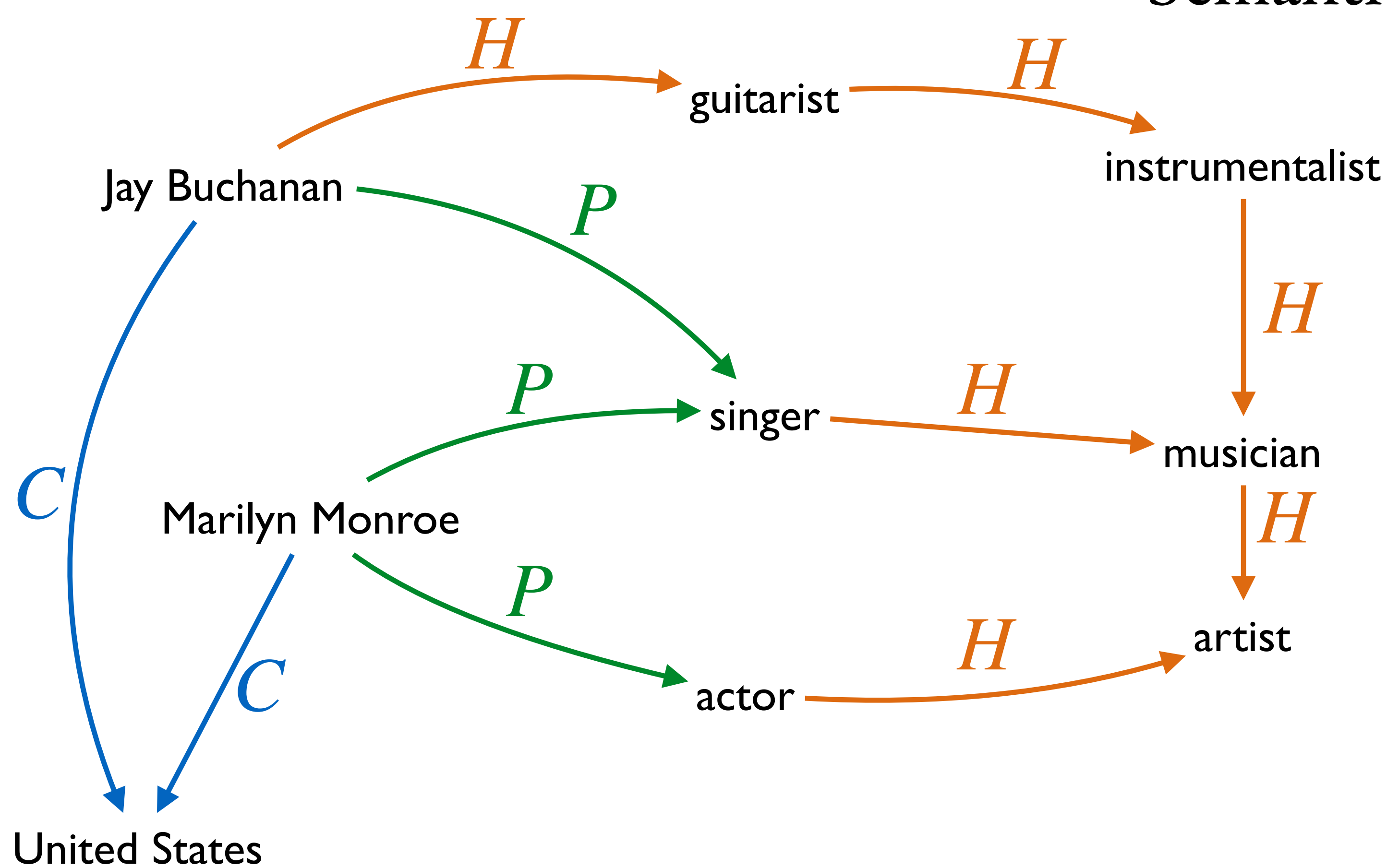
RPQ

$x \xrightarrow{r} y$



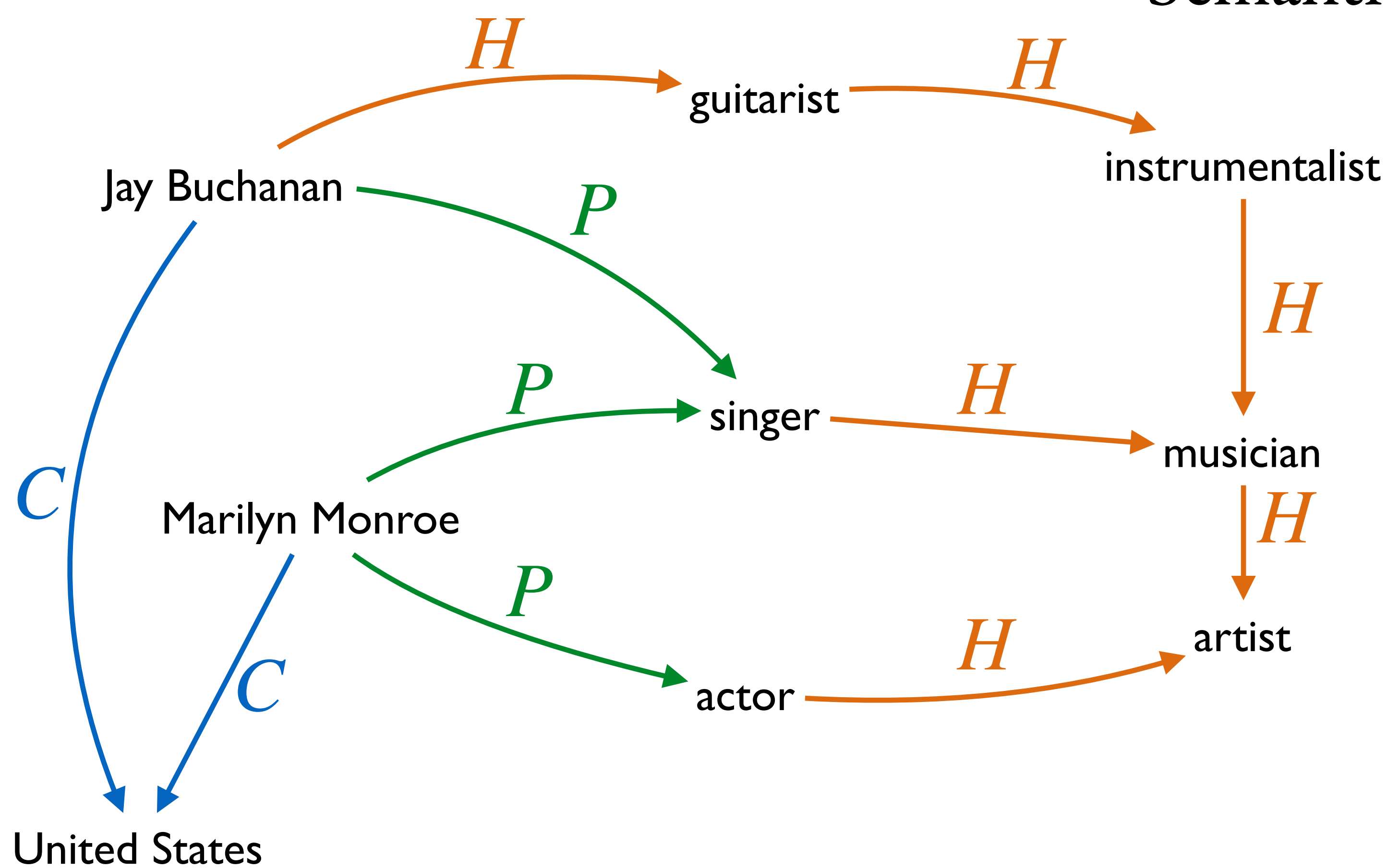
(u,v) is returned iff
there is a path
from u to v
whose label matches r

Regular Path Queries
Semantics



The RPQ $x \xrightarrow{H^*} y$ returns:

Regular Path Queries
Semantics

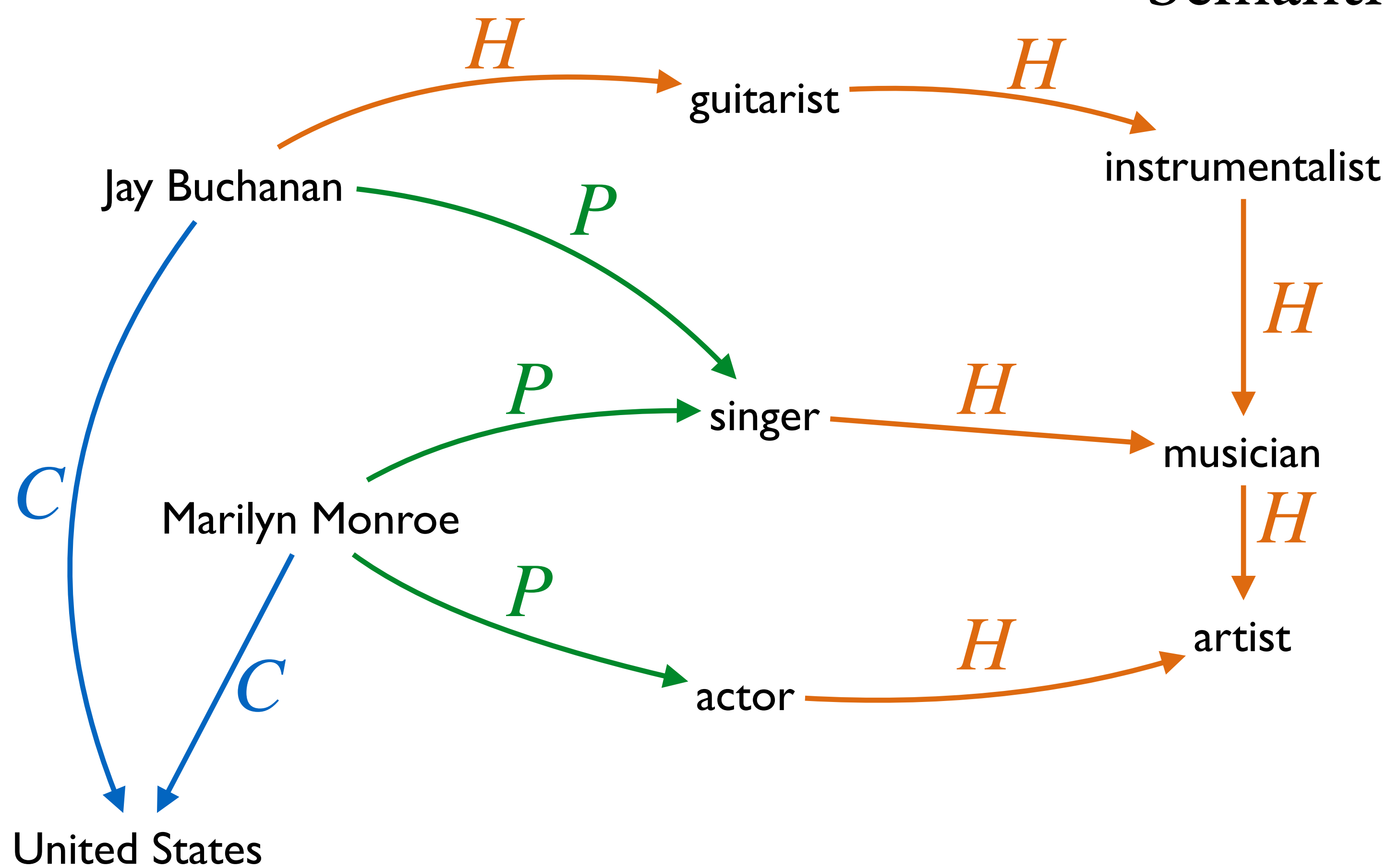


The RPQ $x \xrightarrow{H^*} y$ returns:

(guitarist, guitarist),

Regular Path Queries

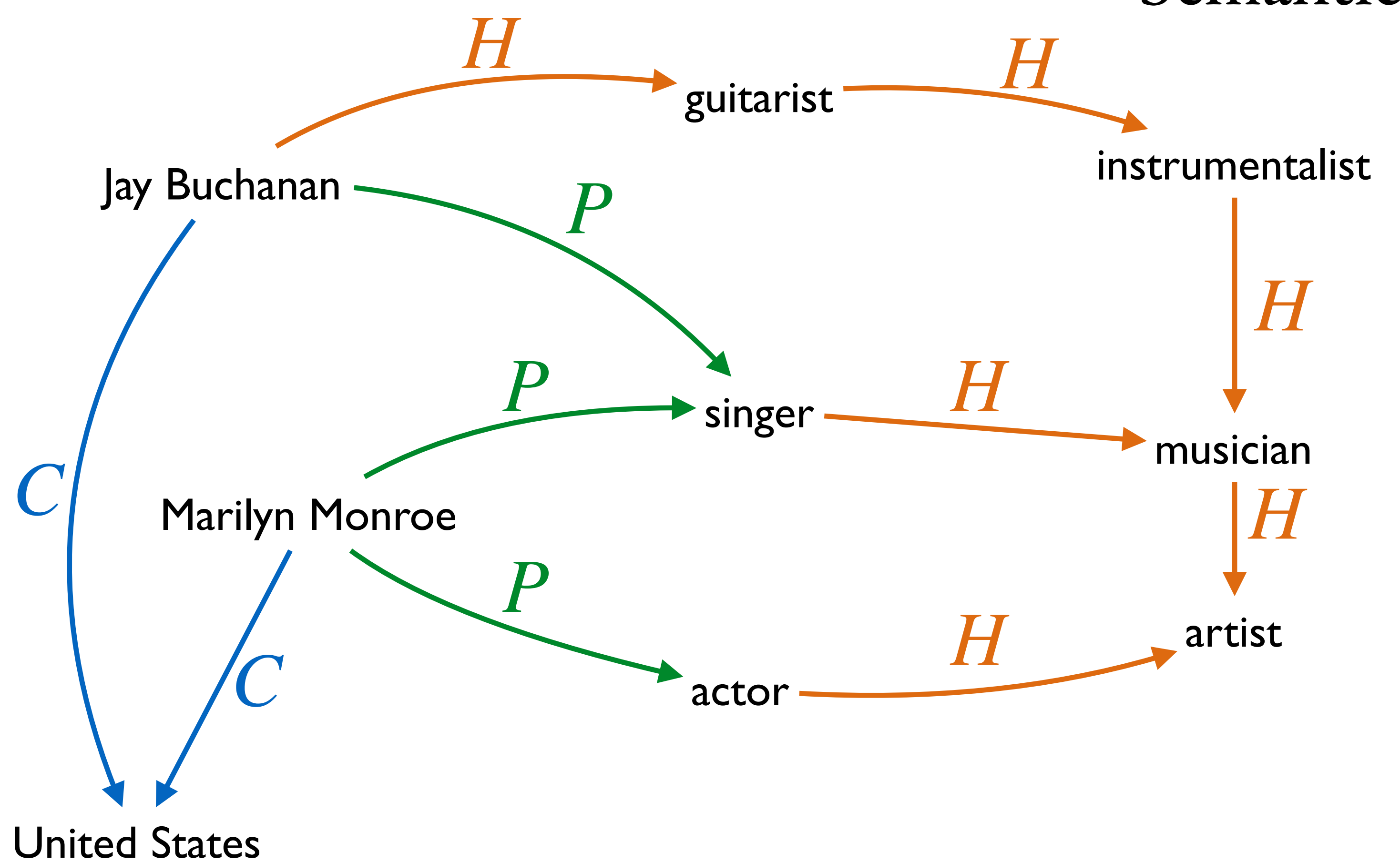
Semantics



The RPQ $x \xrightarrow{H^*} y$ returns: (guitarist, guitarist), (guitarist, instrumentalist),

Regular Path Queries

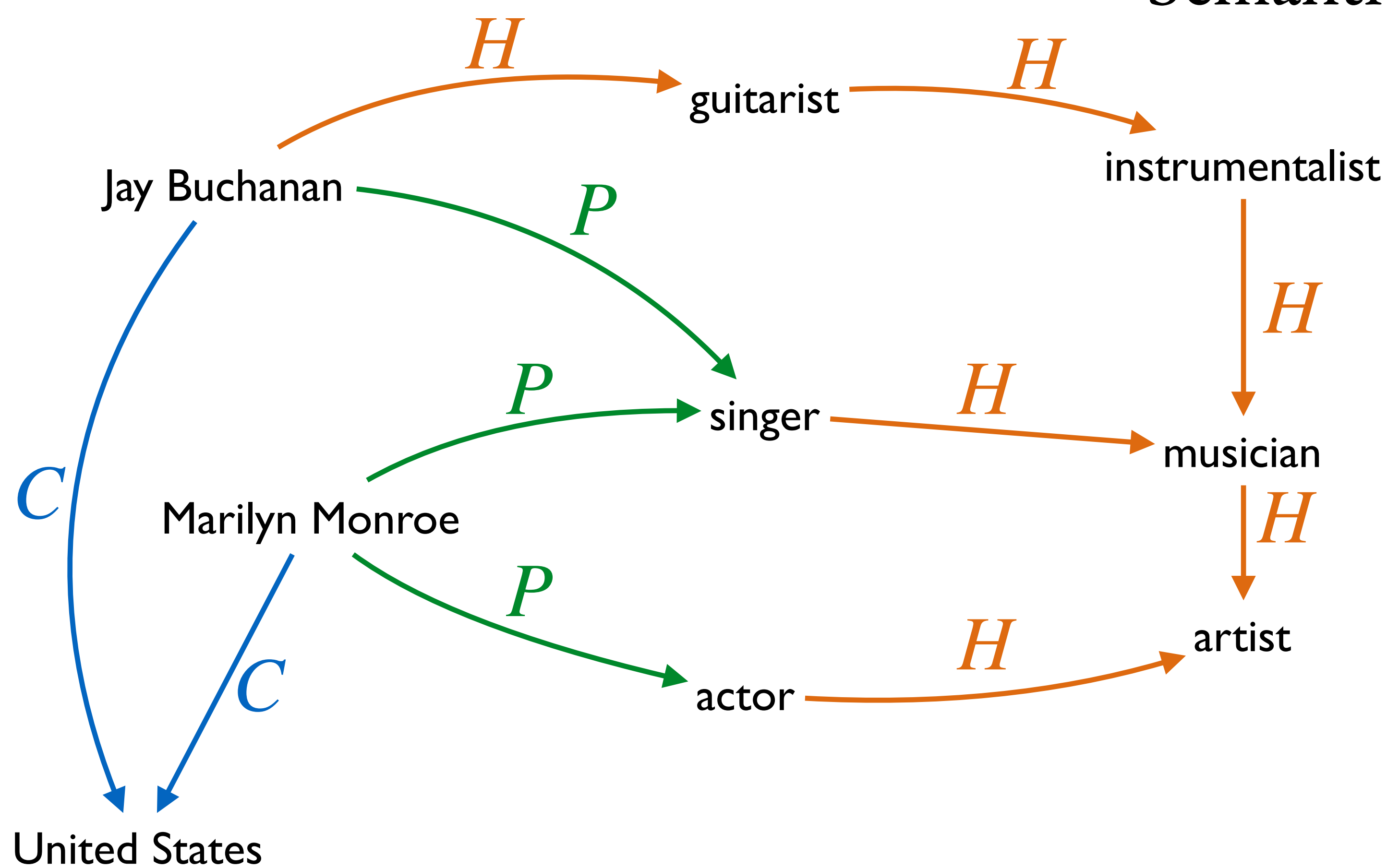
Semantics



The RPQ $x \xrightarrow{H^*} y$ returns:

(guitarist, guitarist), (guitarist, instrumentalist), (guitarist, musician), (guitarist, artist),

Regular Path Queries
Semantics



The RPQ $x \xrightarrow{H^*} y$ returns:

(guitarist, guitarist), (guitarist, instrumentalist), (guitarist, musician), (guitarist, artist), (United States, United States),...

Semantics of RPQs

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

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

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

Semantics of RPQs

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

The **answer of Q on G** 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

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

Semantics of RPQs

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

The **answer of Q on G** is

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

Notation

If $Q = (x \xrightarrow{r} y)$, we sometimes denote $Q(G)$ by $r(G)$

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

Semantics of RPQs (every path semantics)

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

The **answer of Q on G** is

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

Notation

If $Q = (x \xrightarrow{r} y)$, we sometimes denote $Q(G)$ by $r(G)$

Regular Path Queries

Semantics

There are different semantics of RPQs in the literature and in graph database systems!

every path

trail

simple path

shortest path

The differences between these are significant

Regular Path Queries

Semantics

There are different semantics of RPQs in the literature and in graph database systems!

every path

trail

simple path

shortest path

The differences between these are significant

Semantics of RPQs

Why will we consider these different semantics?

Semantics of RPQs

Why will we consider these different semantics?

Each of these semantics is 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

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

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
- Simple path semantics was the first that was studied (back in 1987)

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
- Simple path semantics was the first that was studied (back in 1987)
- Trail semantics is the default in Neo4j Cypher

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
- Simple path semantics was the first that was studied (back in 1987)
- Trail semantics is the default in Neo4j Cypher

What to use in new languages:

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
- Simple path semantics was the first that was studied (back in 1987)
- Trail semantics is the default in Neo4j Cypher

What to use in new languages:

Consensus - all. Every path (walk), shortest, simple, trail.

Semantics of RPQs

Semantics of RPQs (every path semantics)

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

The **answer of Q on G under every path semantics** 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

Semantics of RPQs (every path semantics)

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

The **answer of Q on G under every path semantics** 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

Semantics of RPQs

Semantics of RPQs (every path semantics)

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

The **answer of Q on G under every path semantics** is

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

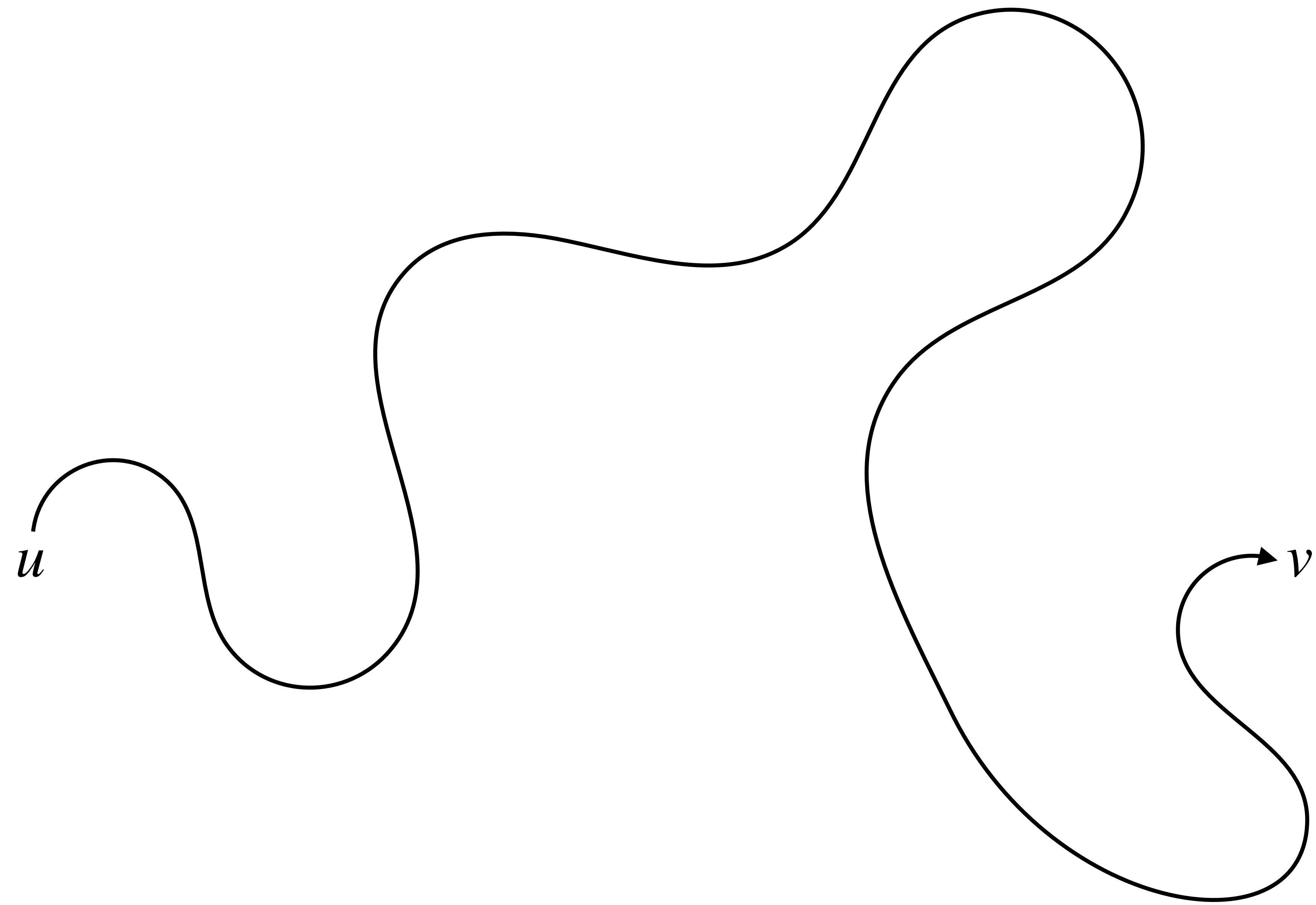
Notice that we do not have any constraint on the path p

Hence, "every path" is eligible for the query

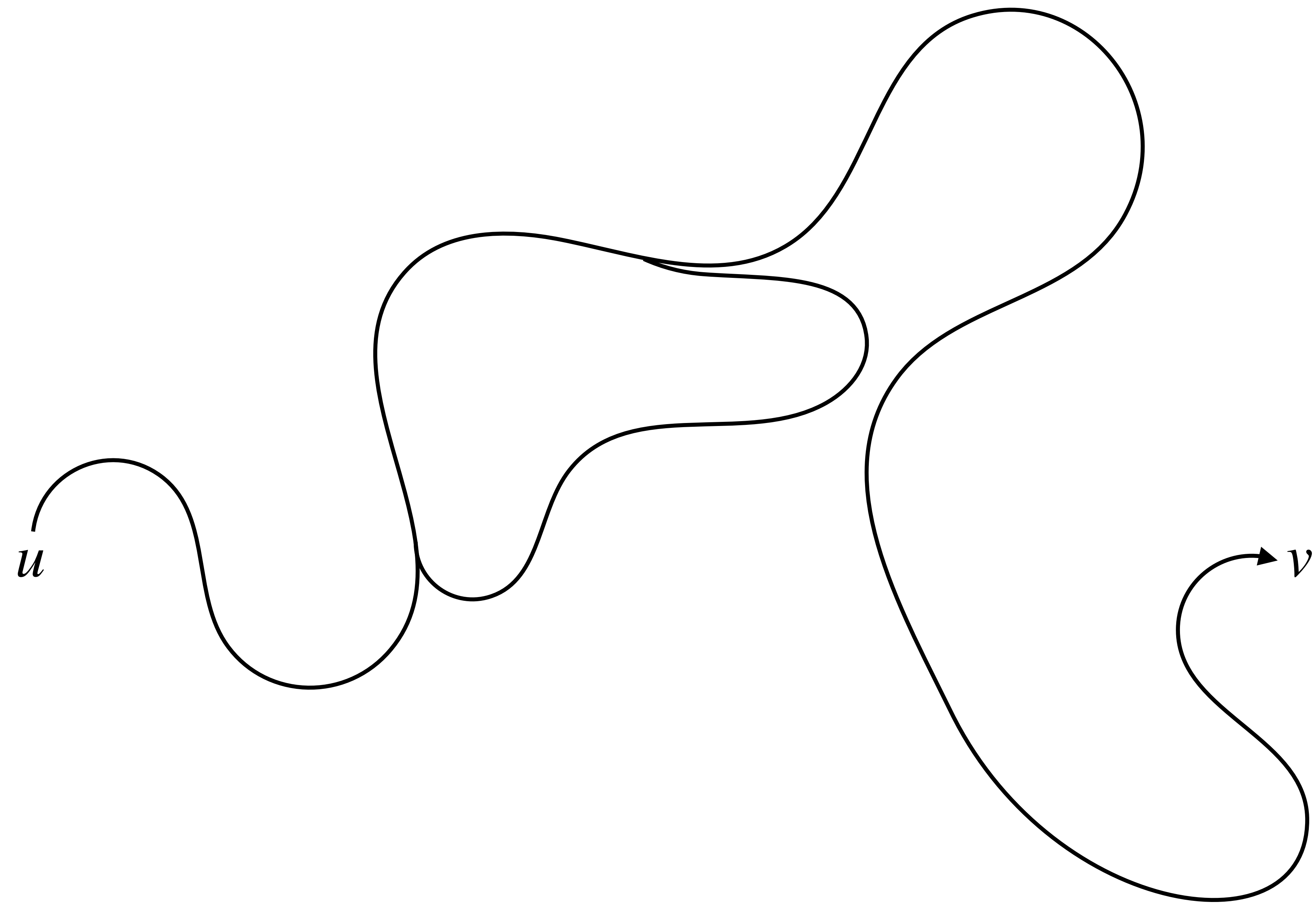
Simple Paths and Trails

u v

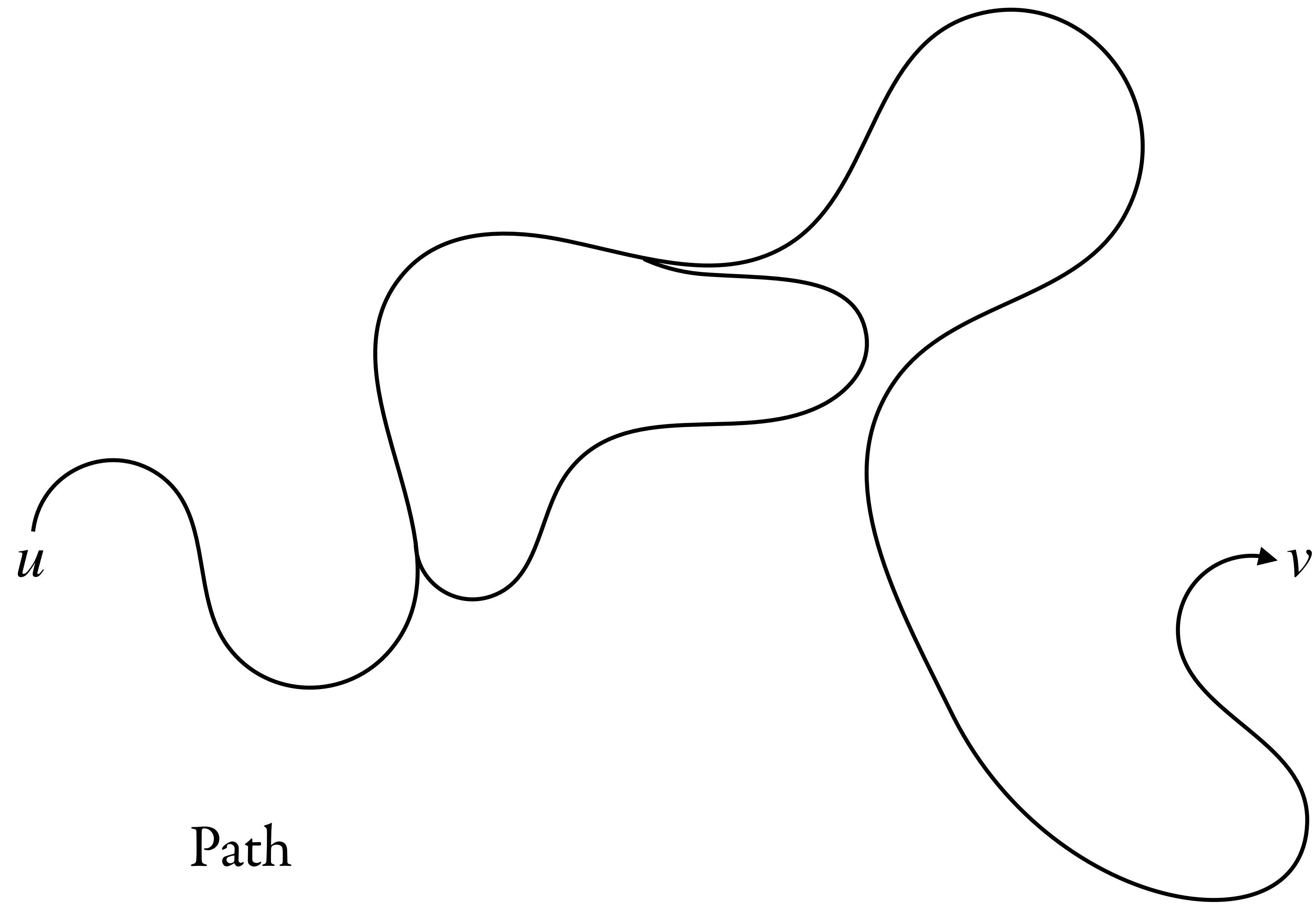
Simple Paths and Trails



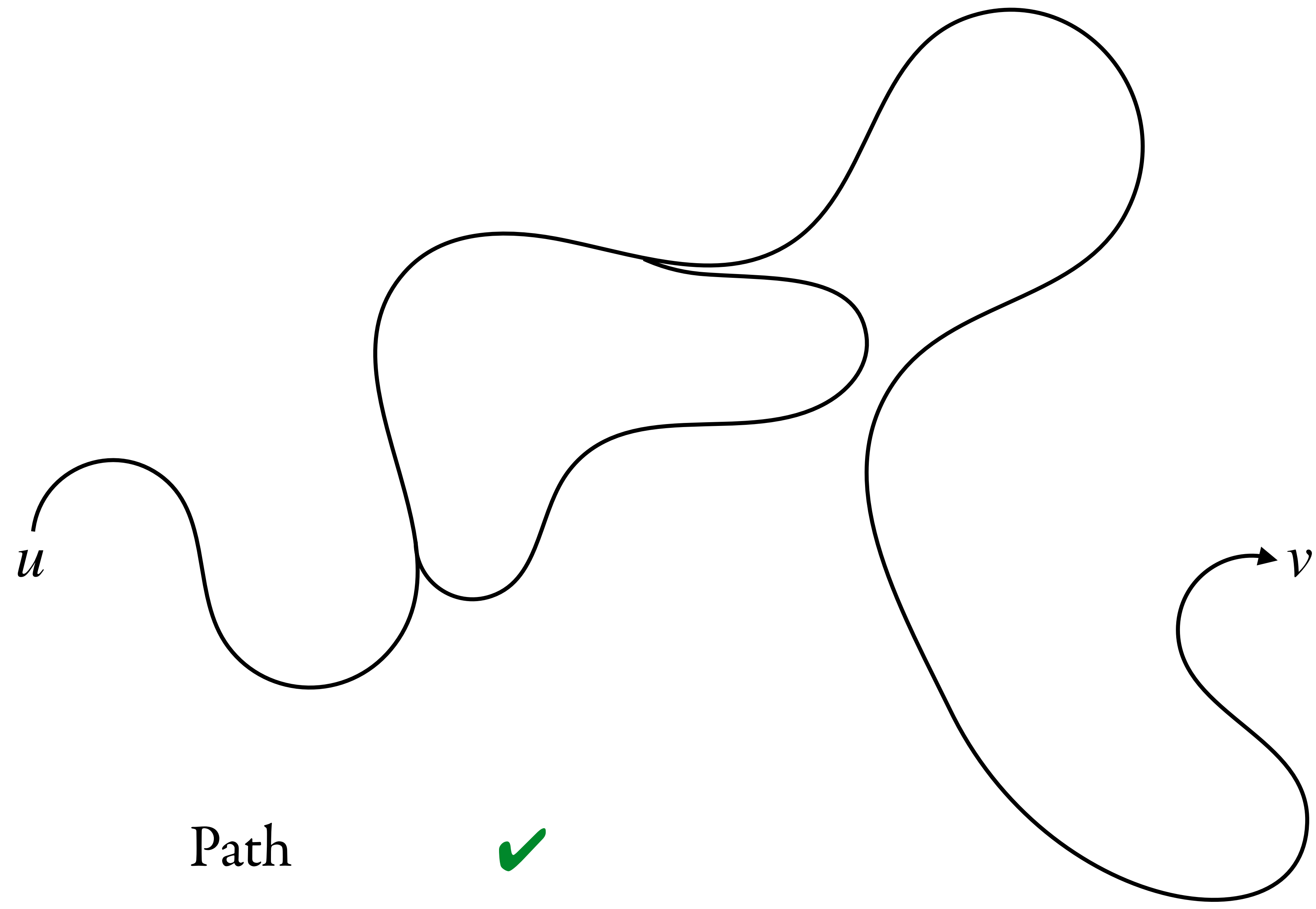
Simple Paths and Trails



Simple Paths and Trails

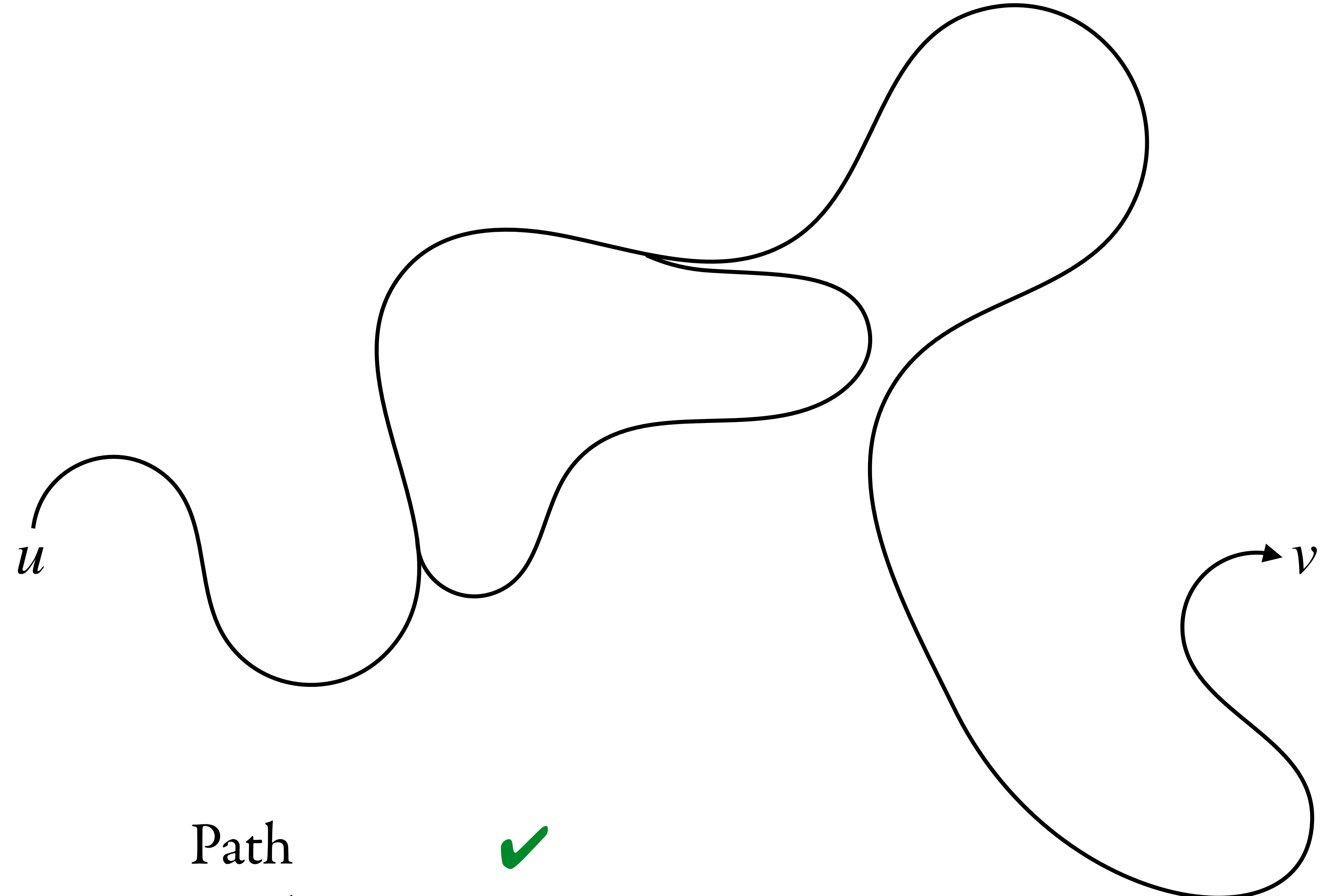


Simple Paths and Trails



Path ✔

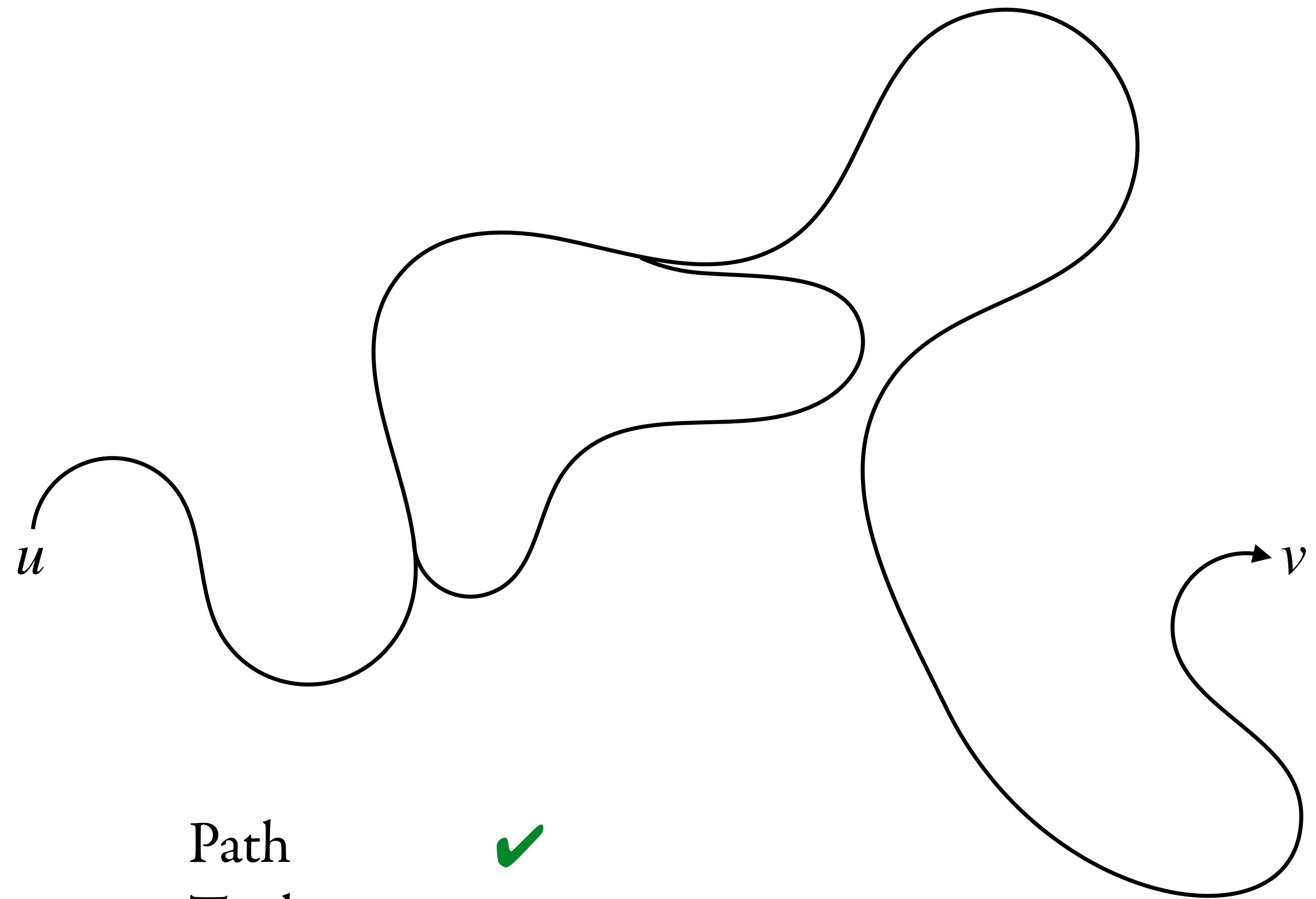
Simple Paths and Trails



Path ✔

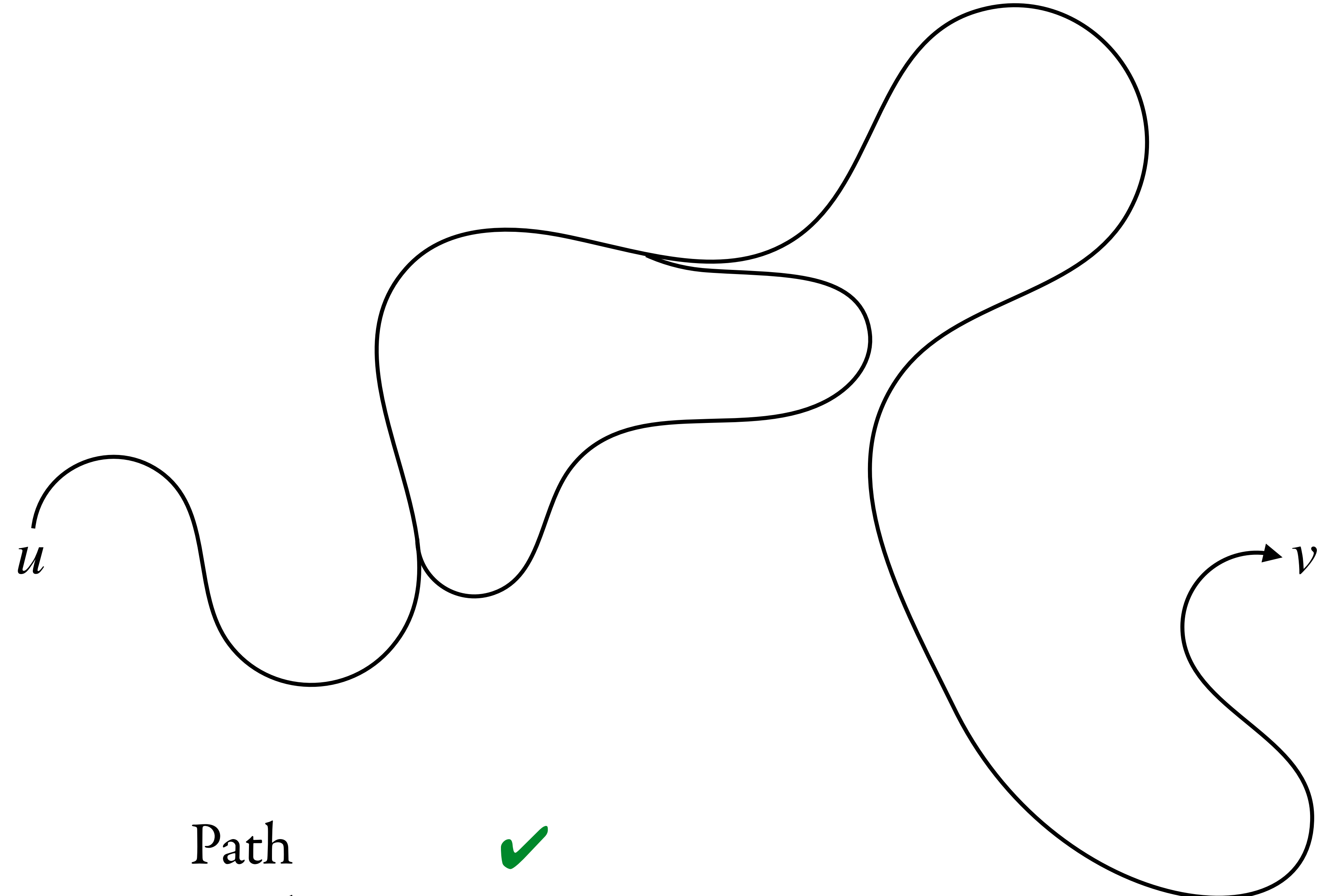
Trail

Simple Paths and Trails



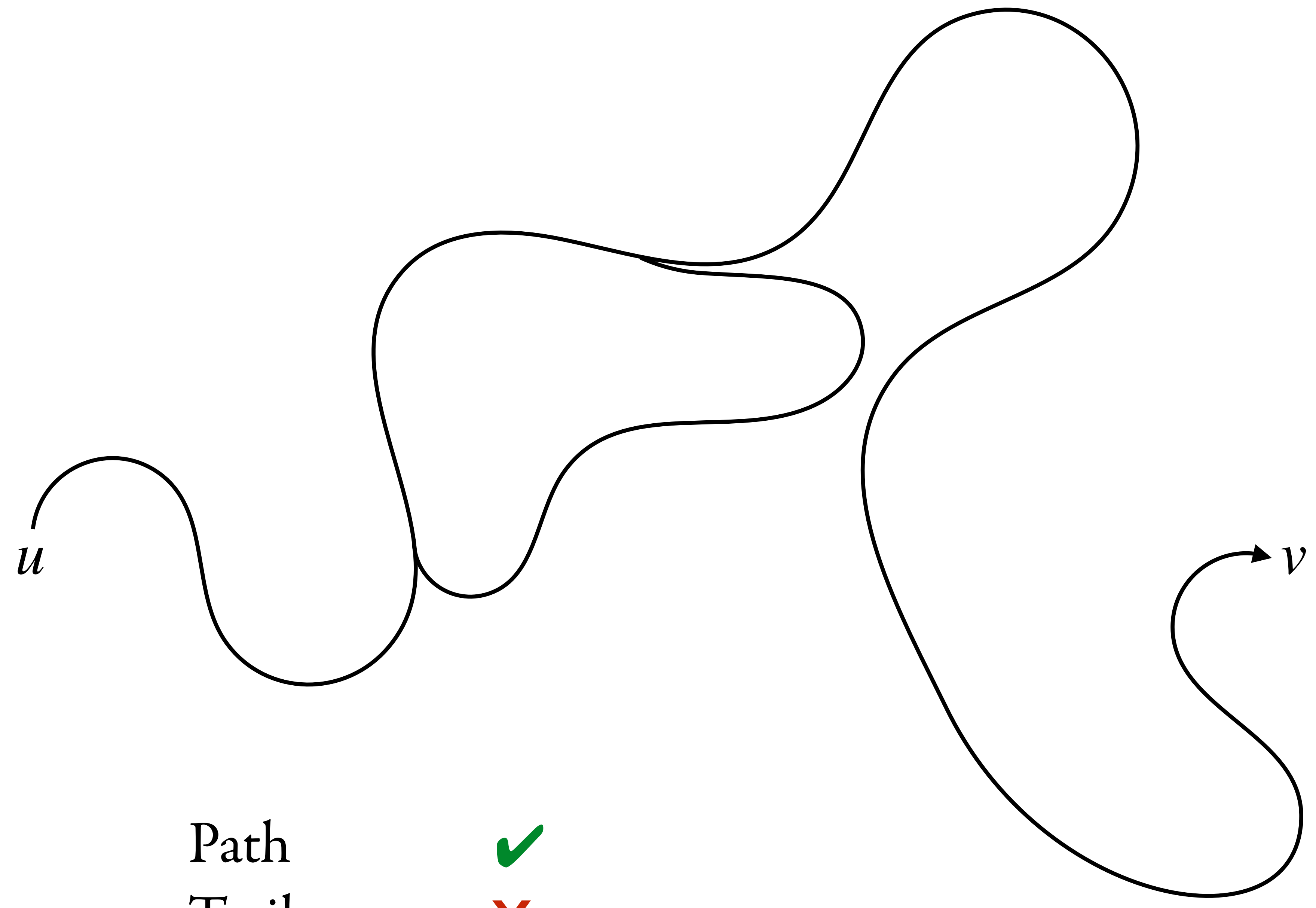
Path ✔
Trail ✘

Simple Paths and Trails



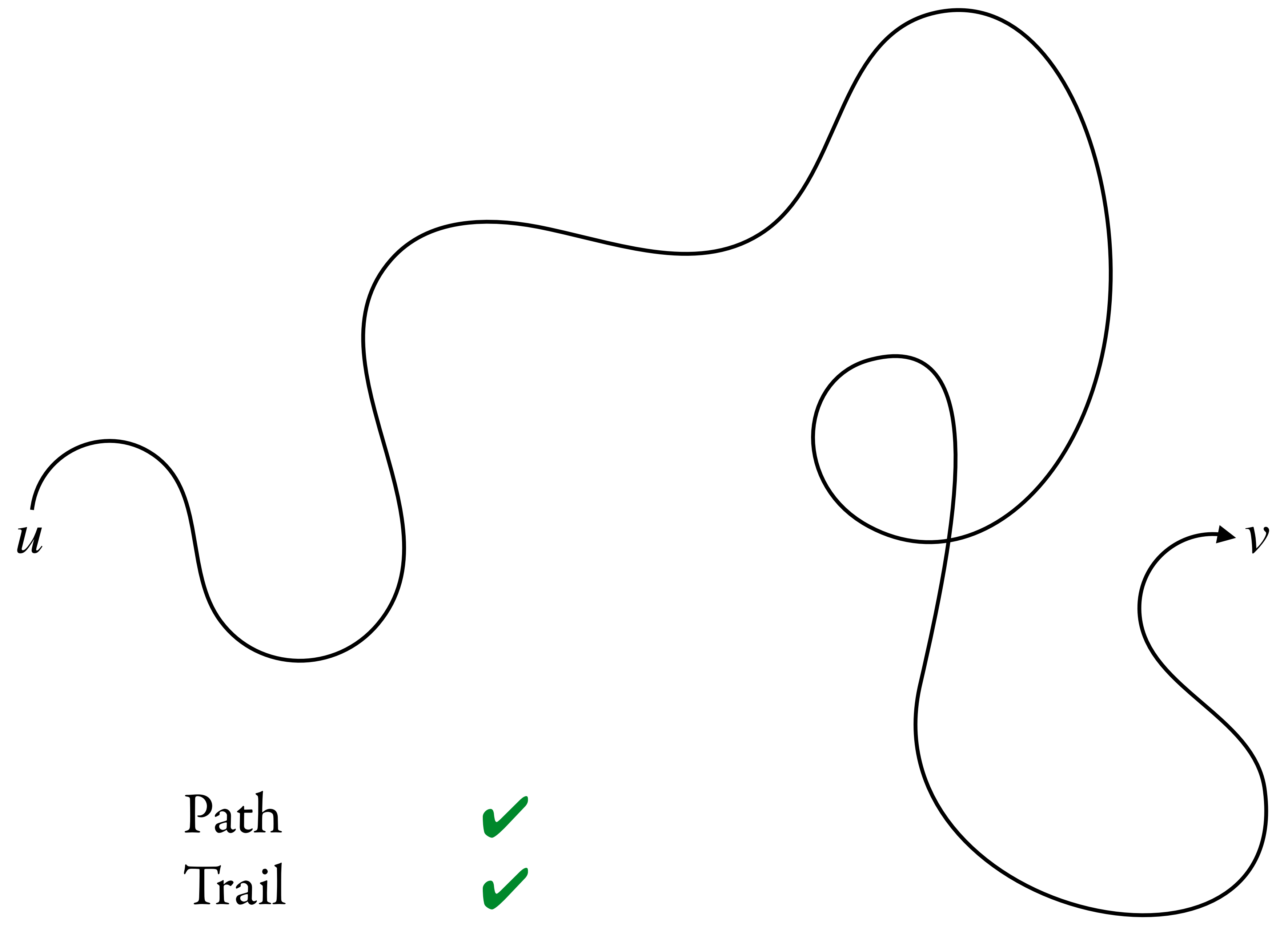
Path ✔
Trail ✘
Simple path

Simple Paths and Trails



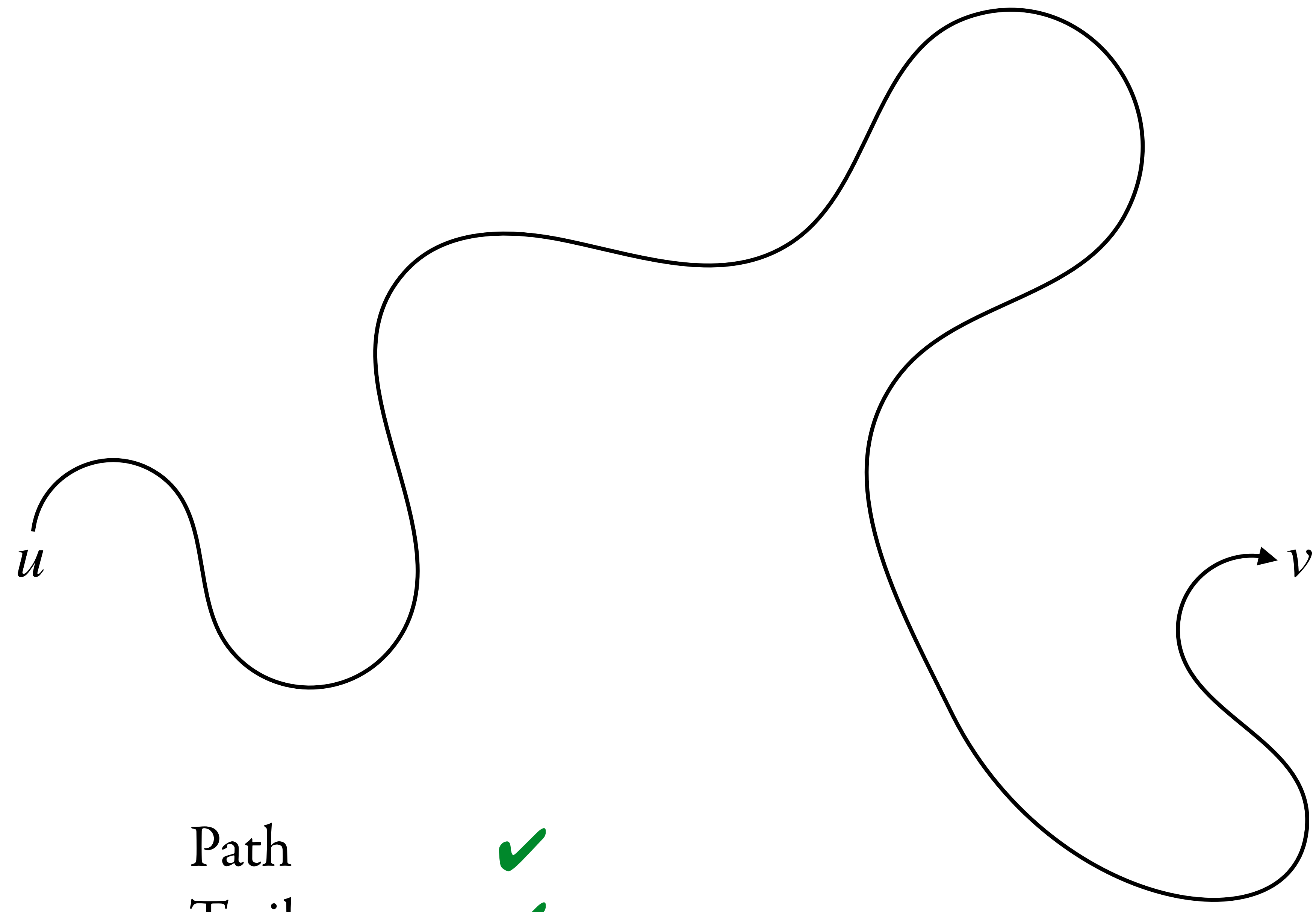
Path ✔
Trail ✘
Simple path ✘

Simple Paths and Trails



Path ✓
Trail ✓
Simple path ✗

Simple Paths and Trails



Path ✔
Trail ✔
Simple path ✔

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 it is empty or

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

Path p is a **trail** if it is empty or

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

Semantics of RPQs

Semantics of RPQs (simple path semantics)

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

The **answer of Q on G under simple path semantics** is

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

Semantics of RPQs

Semantics of RPQs (trail semantics)

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

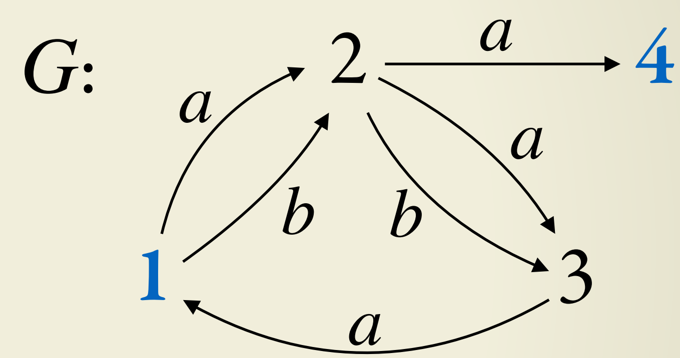
The **answer of Q on G under trail semantics** is

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

RPQ Semantics: Examples

Take $r = (aa)^*$

Take $r = (aa)^*a$



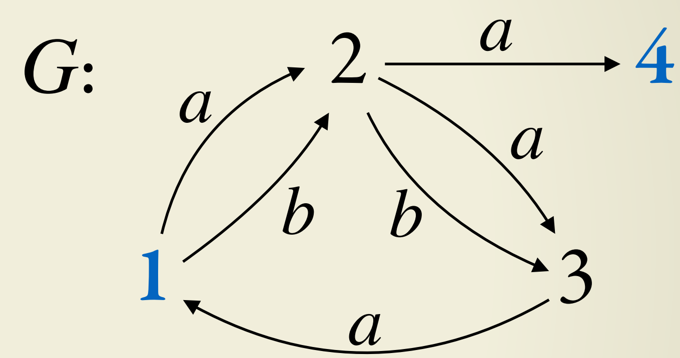
Take $r = (ab)^*a$

RPQ Semantics: Examples

Take $r = (aa)^*$

then $(1,4) \in r(G), r(G)_t,$ and $r(G)_s$

Take $r = (aa)^*a$



Take $r = (ab)^*a$

RPQ Semantics: Examples

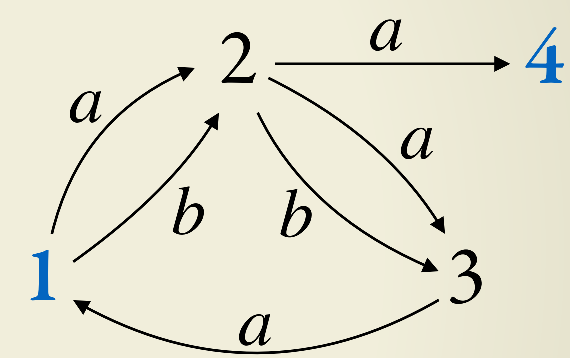
Take $r = (aa)^*$

then $(1,4) \in r(G), r(G)_t$, and $r(G)_s$

Take $r = (aa)^*a$

then $(1,4) \in r(G)$

G:



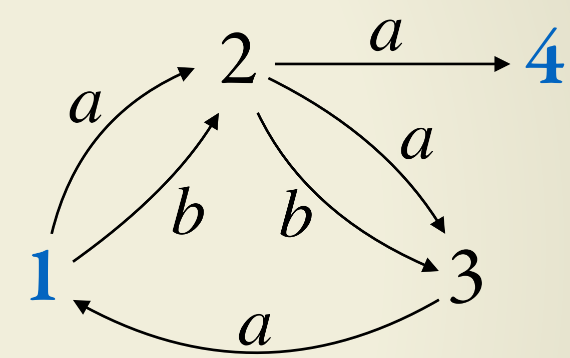
Take $r = (ab)^*a$

RPQ Semantics: Examples

Take $r = (aa)^*$
 then $(1,4) \in r(G)$, $r(G)_t$, and $r(G)_s$

Take $r = (aa)^*a$
 then $(1,4) \in r(G)$
 but $(1,4) \notin r(G)_t$ or $r(G)_s$

G:



Take $r = (ab)^*a$

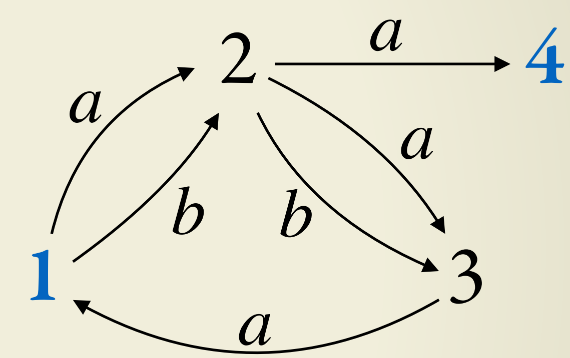
RPQ Semantics: Examples

Take $r = (aa)^*$
then $(1,4) \in r(G)$, $r(G)_t$, and $r(G)_s$

Take $r = (aa)^*a$
then $(1,4) \in r(G)$
but $(1,4) \notin r(G)_t$ or $r(G)_s$

Take $r = (ab)^*a$
then $(1,4) \in r(G)$ and $r(G)_t$

G:



RPQ Semantics: Examples

Take $r = (aa)^*$

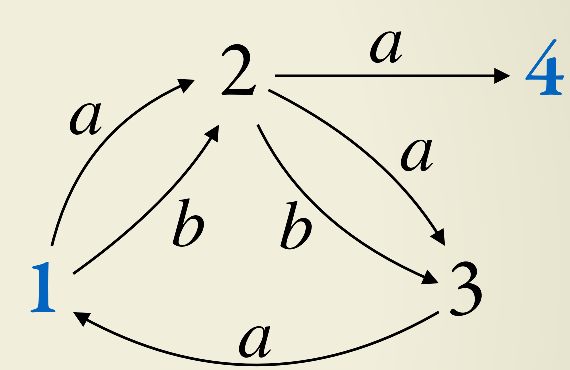
then $(1,4) \in r(G)$, $r(G)_t$, and $r(G)_s$

Take $r = (aa)^*a$

then $(1,4) \in r(G)$

but $(1,4) \notin r(G)_t$ or $r(G)_s$

G:



Take $r = (ab)^*a$

then $(1,4) \in r(G)$ and $r(G)_t$

but $(1,4) \notin r(G)_s$

Conjunctive Regular Path Queries

Definition (Conjunctive Regular Path Query)

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

$$Q(\bar{x}) := \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$$

where

- $\bar{x}$ is a tuple of variables from $\{y_1, \ldots, y_n, z_1, \ldots, z_n\}$ and
- $(y_i \xrightarrow{r_i} z_i)$ is an RPQ over Σ for all $i \in [n]$

Conjunctive Regular Path Queries

Definition (Conjunctive Regular Path Query)

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

$$Q(\bar{x}) := \left((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\right)$$

where

- $\bar{x}$ is a tuple of variables from $\{y_1, \ldots, y_n, z_1, \ldots, z_n\}$ and
- $(y_i \xrightarrow{r_i} z_i)$ is an RPQ over Σ for all $i \in [n]$

Observation 1

Since every symbol a in Σ is a regular expression,

every CQ over graphs is also a CRPQ

Conjunctive Regular Path Queries

Definition (Conjunctive Regular Path Query)

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

$$Q(\bar{x}) := \left((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\right)$$

where

- $\bar{x}$ is a tuple of variables from $\{y_1,\ldots, y_n, z_1,\ldots, z_n\}$ and
- $(y_i \xrightarrow{r_i} z_i)$ is an RPQ over Σ for all $i \in [n]$

Observation 1

Essentially a CQ where building blocks are RPQs

Observation 2

Since every symbol a in Σ is a regular expression,
every CQ over graphs is also a CRPQ

Conjunctive Regular Path Queries

Semantics of CRPQs (every path semantics)

Let $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be a CRPQ and $G = (V, E)$ be a graph

The set of **answers of Q on G** (under **every path** semantics) is

$Q(G) = \{\, \mathrm{h}(\bar{x}) \mid \mathrm{h}$ is a homomorphism from vars(Q) to V

such that $(\mathrm{h}(y_i), \mathrm{h}(z_i)) \in r_i(G)$ for every $i \in [n]\}$

Conjunctive Regular Path Queries

Semantics of CRPQs (every path semantics)

Let $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be a CRPQ and $G = (V, E)$ be a graph

The set of **answers of Q on G** (under **every path** semantics) is

$Q(G) = \{\, \mathrm{h}(\bar{x}) \mid \mathrm{h}$ is a homomorphism from vars(Q) to V

such that $(\mathrm{h}(y_i), \mathrm{h}(z_i)) \in r_i(G)$ for every $i \in [n]\}$

Answers of Q on G under **simple path** and **trail** semantics are defined analogously: we require that

$(\mathrm{h}(x_i), \mathrm{h}(y_i)) \in r_i(G)_s$ and

$(\mathrm{h}(x_i), \mathrm{h}(y_i)) \in r_i(G)_t$ respectively

Conjunctive Regular Path Queries

Semantics of CRPQs (every path semantics)

Let $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be a CRPQ and $G = (V, E)$ be a graph

The set of **answers of Q on G** (under **every path** semantics) is

$Q(G) = \{\, \mathrm{h}(\bar{x}) \mid \mathrm{h}$ is a homomorphism from vars(Q) to V

such that $(\mathrm{h}(y_i), \mathrm{h}(z_i)) \in r_i(G)$ for every $i \in [n]\}$

Answers of Q on G under **simple path** and **trail** semantics are defined analogously: we require that

$(\mathrm{h}(x_i), \mathrm{h}(y_i)) \in r_i(G)_s$ and

$(\mathrm{h}(x_i), \mathrm{h}(y_i)) \in r_i(G)_t$ respectively

Notation: $Q(G)_s$ for simple path semantics

$Q(G)_t$ for trail semantics

Conjunctive Queries (CQs)

Query Evaluation

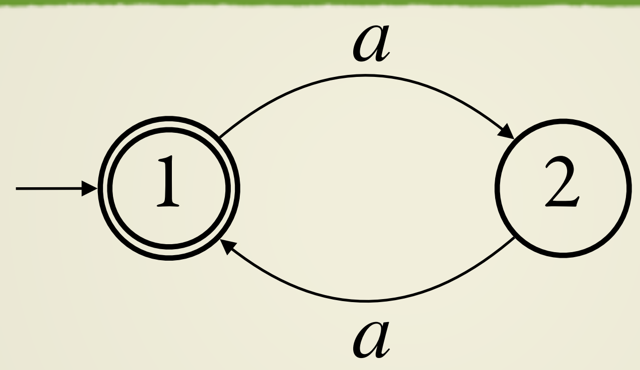
Regular Path Queries (RPQs)

Conjunctive Regular Path Queries (CRPQs)

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 (or "final") states

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

In the example:

$S = \{1,2\}$

$A = \{a\}$

$\delta = \{(1,a,2), (2,a,1)\}$

$I = \{1\}$

$F = \{1\}$

Evaluation Problems

RPQ Evaluation (every path semantics)

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

Question: Is $(u, v) \in Q(G)$?

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 $\bar{u}$ of nodes
conjunctive regular path query Q

Question: Is $\bar{u} \in Q(G)$?

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 $\bar{u}$ of nodes
conjunctive regular path query Q

Question: Is $\bar{u} \in Q(G)$?

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

RPQs, Every Path Semantics

Theorem

RPQ Evaluation under every path semantics is in PTIME

RPQs, Every Path Semantics

Theorem

RPQ Evaluation under every path semantics is in PTIME

Proof (sketch)

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

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

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)

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

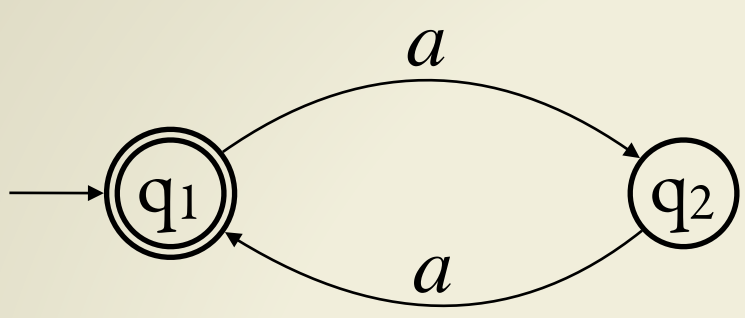
(This is similar to a product between automata)

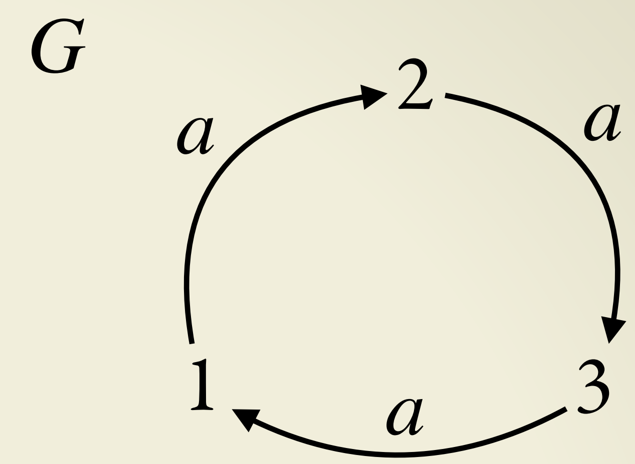
Accept iff there is a path from (i,u) to (f,v) in $G \times N$, for some $i \in I$ and $f \in F$

Example

RPQ Evaluation under Every Path Semantics

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



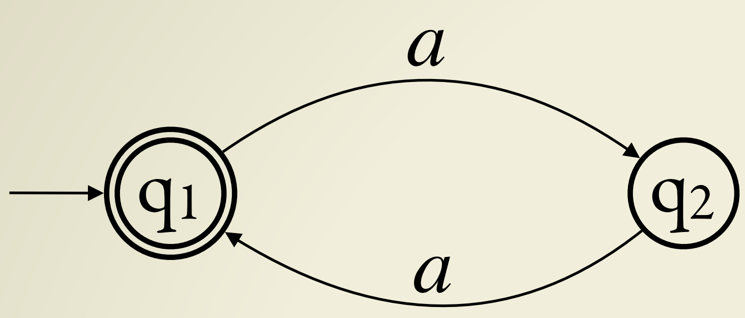


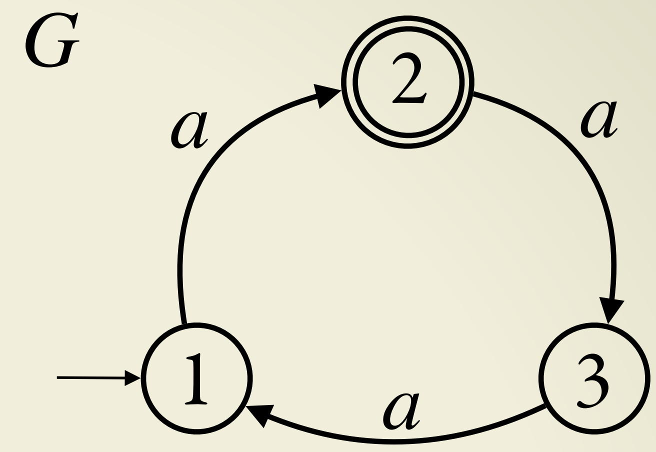
Is (1,2) in $r(G)$?

Example

RPQ Evaluation under Every Path Semantics

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



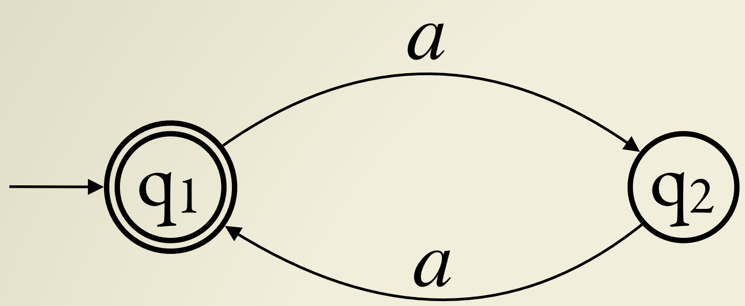


Is (1,2) in $r(G)$?

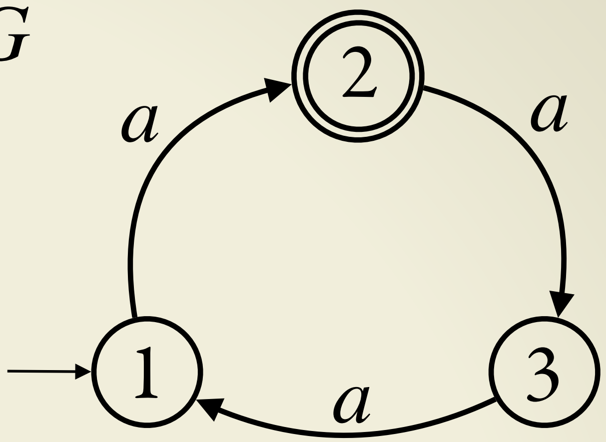
Example

RPQ Evaluation under Every Path Semantics

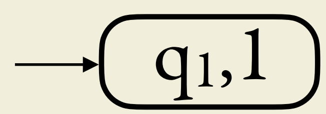
Consider the RPQ $r = (aa)^*$



G



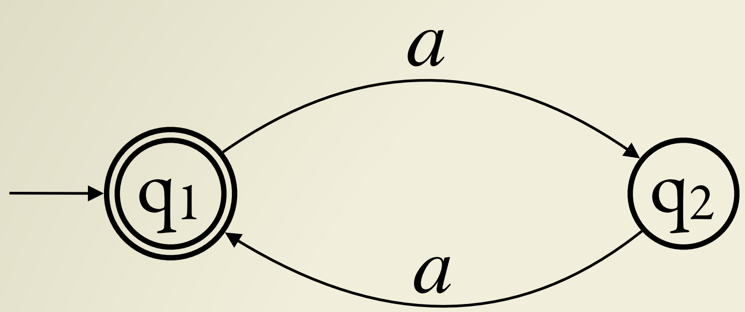
Is (1,2) in $r(G)$?



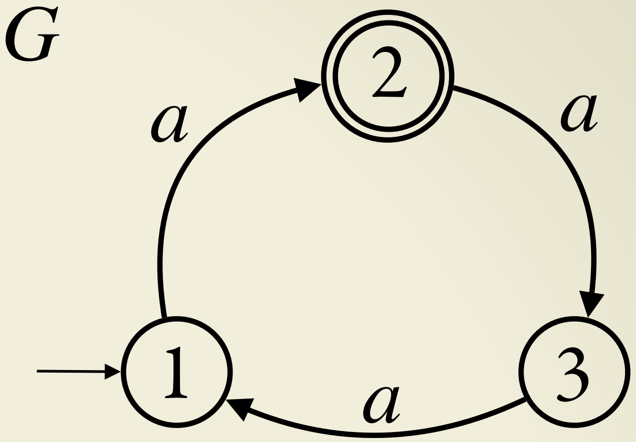
Example

RPQ Evaluation under Every Path Semantics

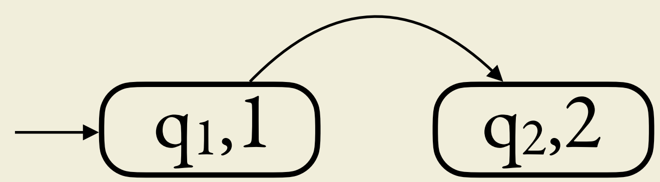
Consider the RPQ $r = (aa)^*$



G



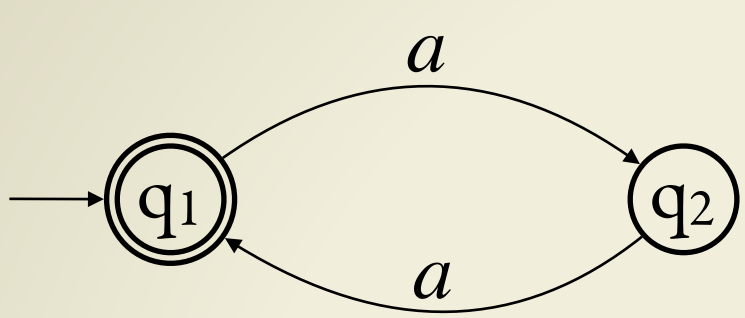
Is (1,2) in $r(G)$?



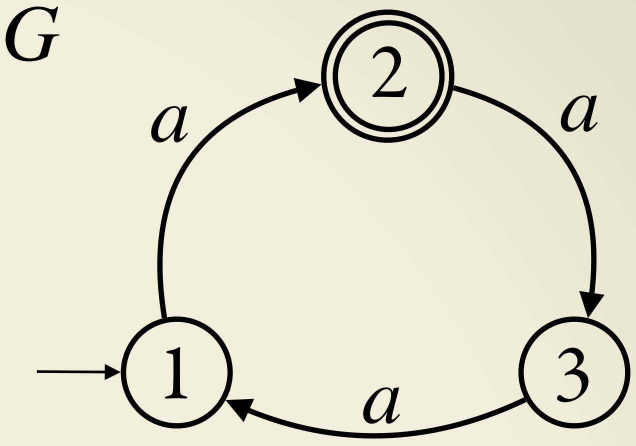
Example

RPQ Evaluation under Every Path Semantics

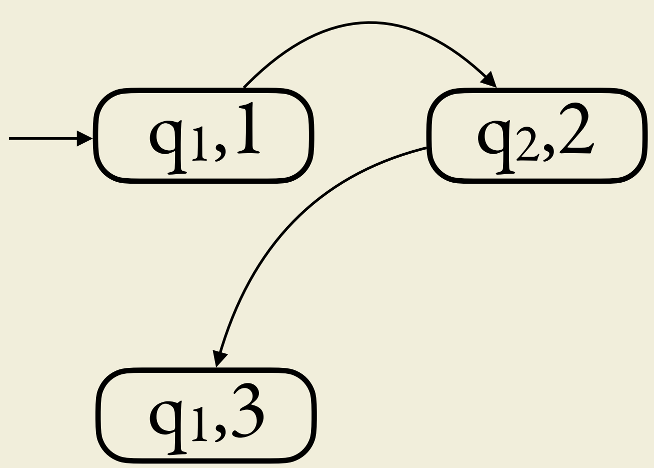
Consider the RPQ $r = (aa)^*$



G



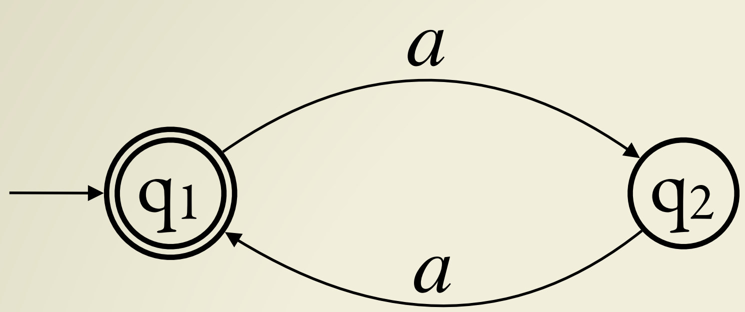
Is (1,2) in $r(G)$?



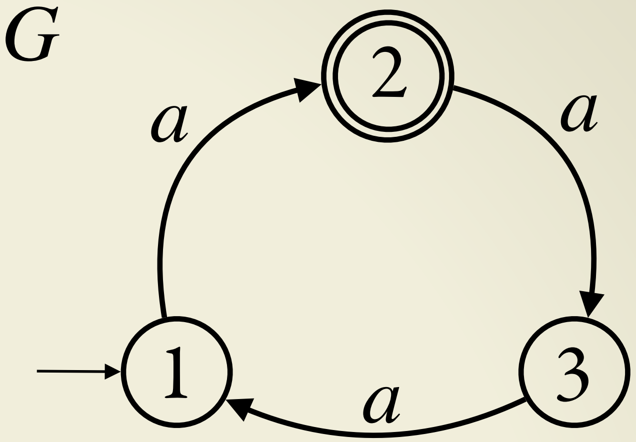
Example

RPQ Evaluation under Every Path Semantics

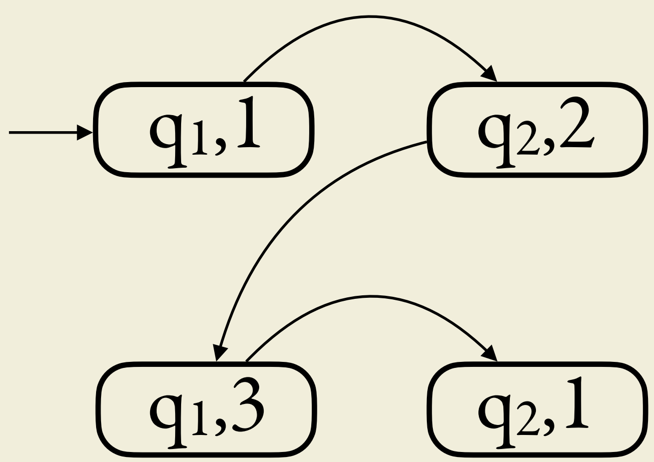
Consider the RPQ $r = (aa)^*$



G



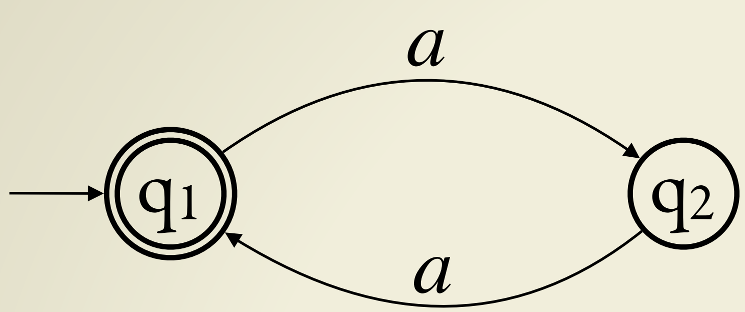
Is (1,2) in $r(G)$?



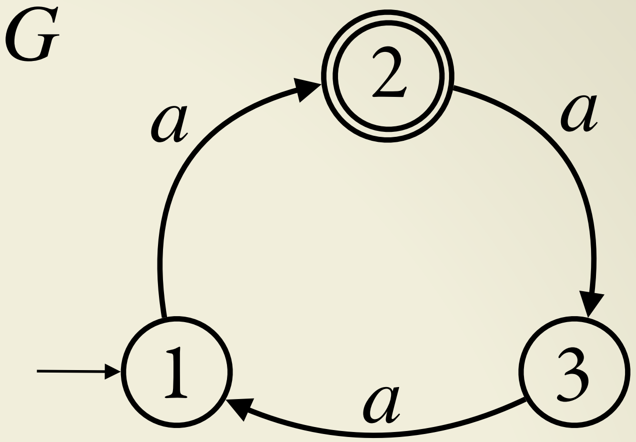
Example

RPQ Evaluation under Every Path Semantics

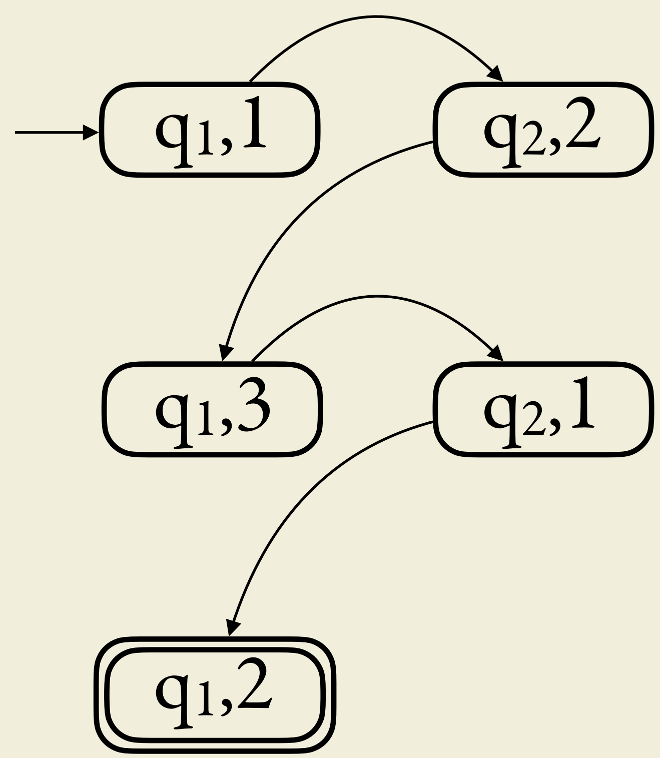
Consider the RPQ $r = (aa)^*$



G



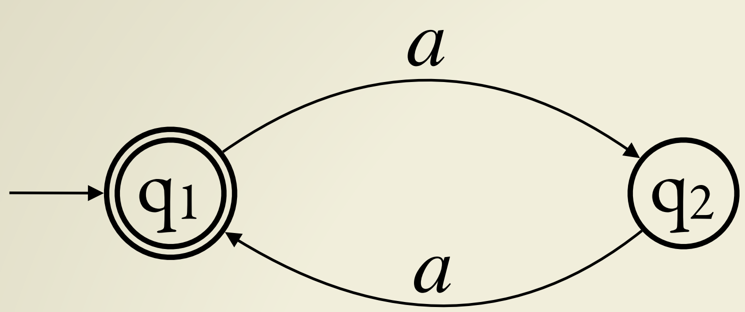
Is (1,2) in $r(G)$?



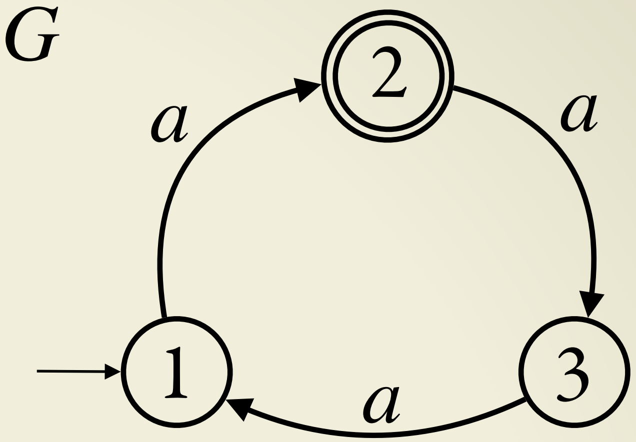
Example

RPQ Evaluation under Every Path Semantics

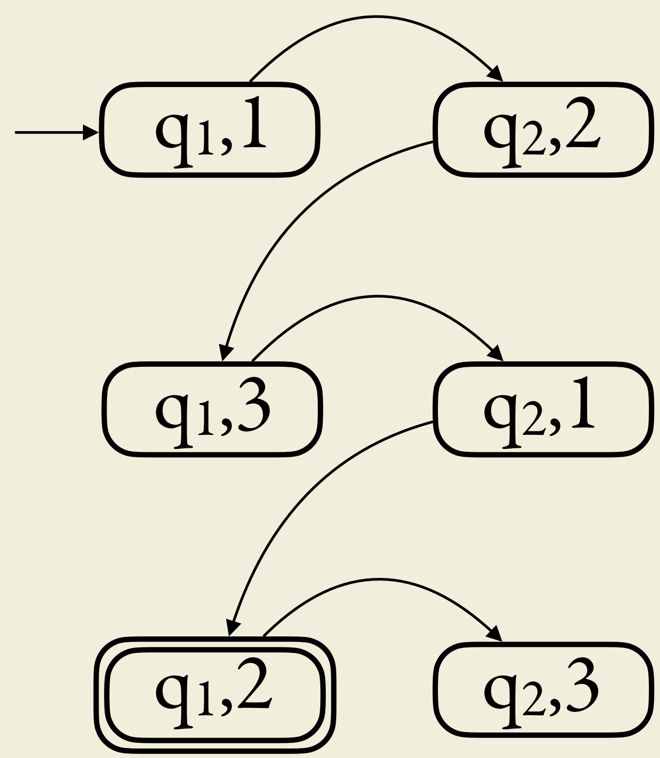
Consider the RPQ $r = (aa)^*$



G



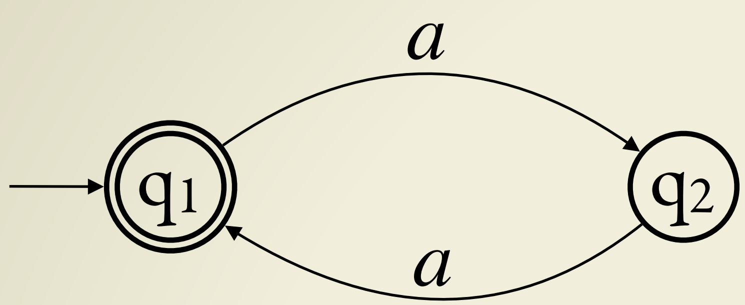
Is (1,2) in $r(G)$?



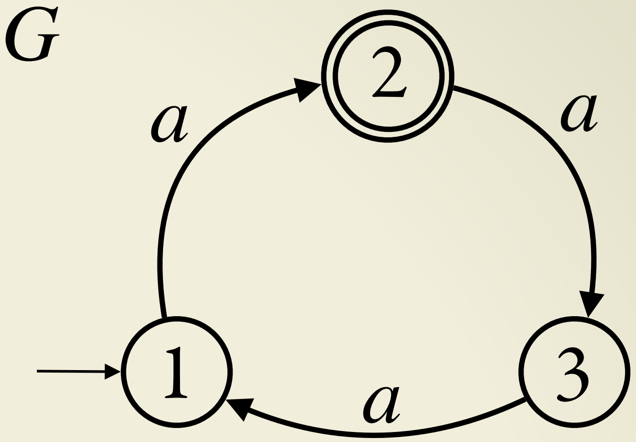
Example

RPQ Evaluation under Every Path Semantics

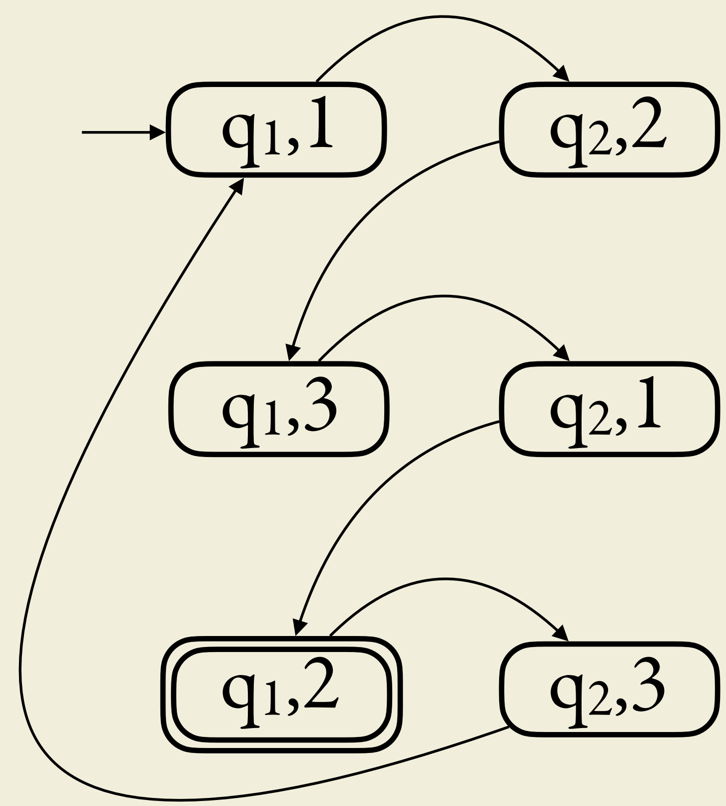
Consider the RPQ $r = (aa)^*$



G



Is (1,2) in $r(G)$?



RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

Lower bound:

Reduction from (directed) Hamiltonian Path

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

Lower bound:

Reduction from (directed) Hamiltonian Path

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

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

Lower bound:

Reduction from (directed) Hamiltonian Path

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

Let G_a be obtained from H by labeling each edge with a

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

Lower bound:

Reduction from (directed) Hamiltonian Path

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

Let G_a be obtained from H by labeling each edge with a

Then H has a Hamiltonian Path from u to v

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-complete

Proof (sketch)

Input: $Q = (x \xrightarrow{r} y)$, graph data G, and pair of nodes (u, v)

Upper bound:

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

Lower bound:

Reduction from (directed) Hamiltonian Path

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

Let G_a be obtained from H by labeling each edge with a

Then H has a Hamiltonian Path from u to v

iff (u,v) in $Q(G_a)_s$ with $Q = (x \xrightarrow{a^{n-1}} y)$

RPQs, Simple Path Semantics

Theorem

RPQ Evaluation under simple path semantics is NP-hard
under data complexity

RPQs, Simple Path Semantics

Theorem

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

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 a pair (u,v) of nodes,
is there a simple path of even length from u to v?

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 a pair (u,v) of nodes,
is there a simple path of even length from u to v?

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

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 a pair (u,v) of nodes,

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 $(u, v) \in Q(G_a)_s$

RPQs, Simple Path Semantics

Theorem

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

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?

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]

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]

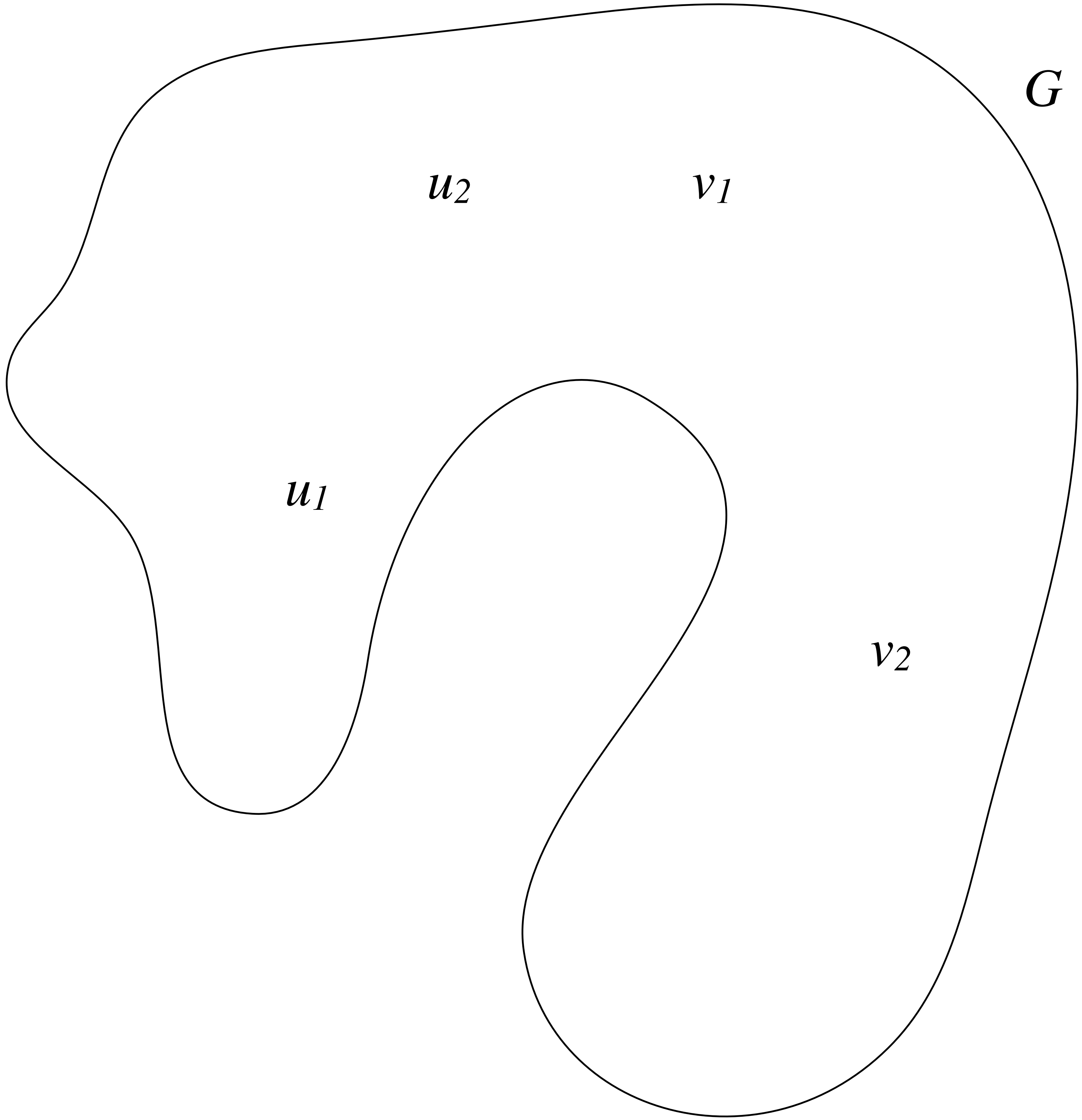
Proof (sketch)

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

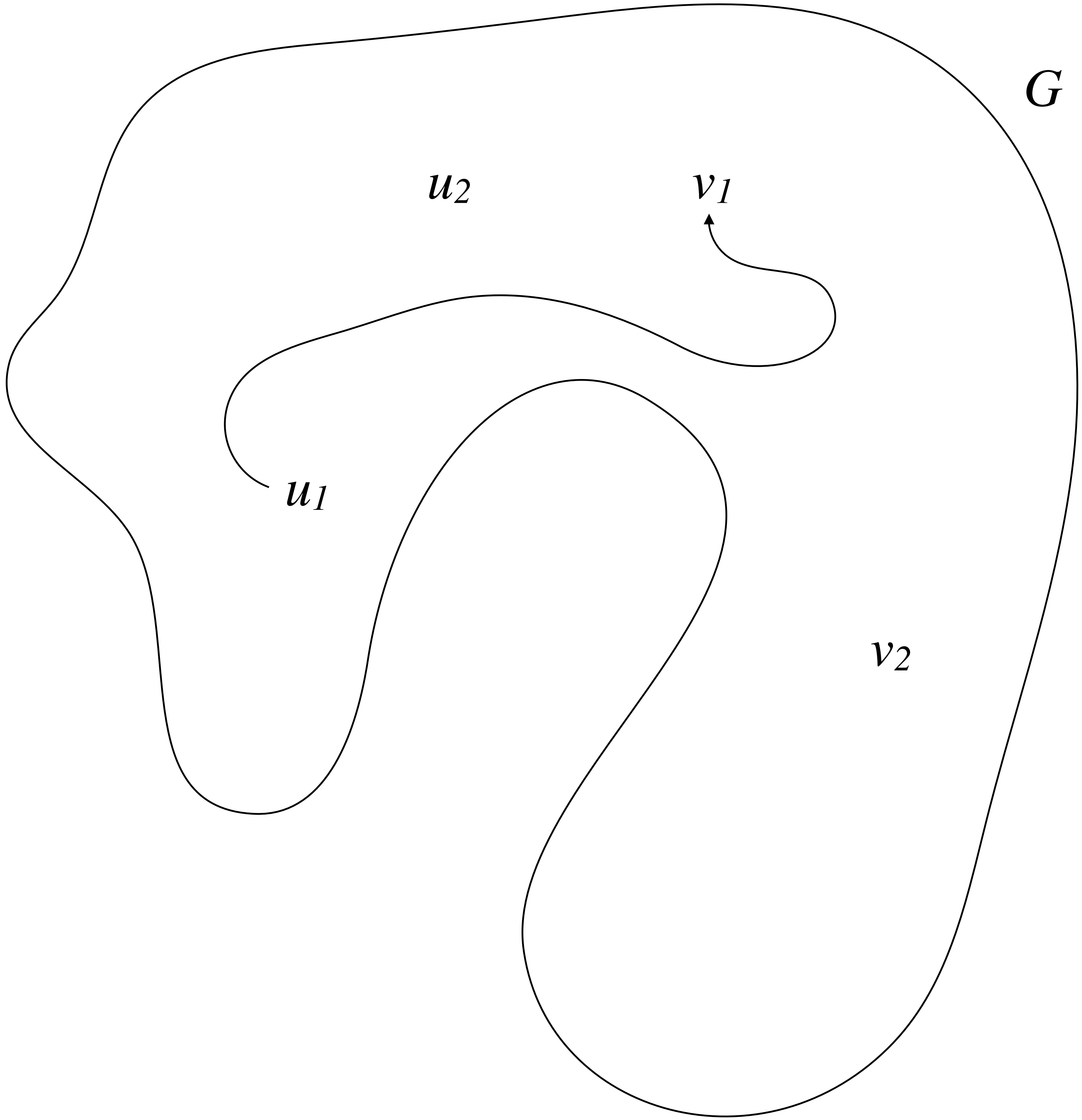
Then G has node-disjoint paths p_1 and p_2, from u_1 to v_1 and from u_2 to v_2 iff

$$(u_1, v_2) \in Q(G_b)_s$$

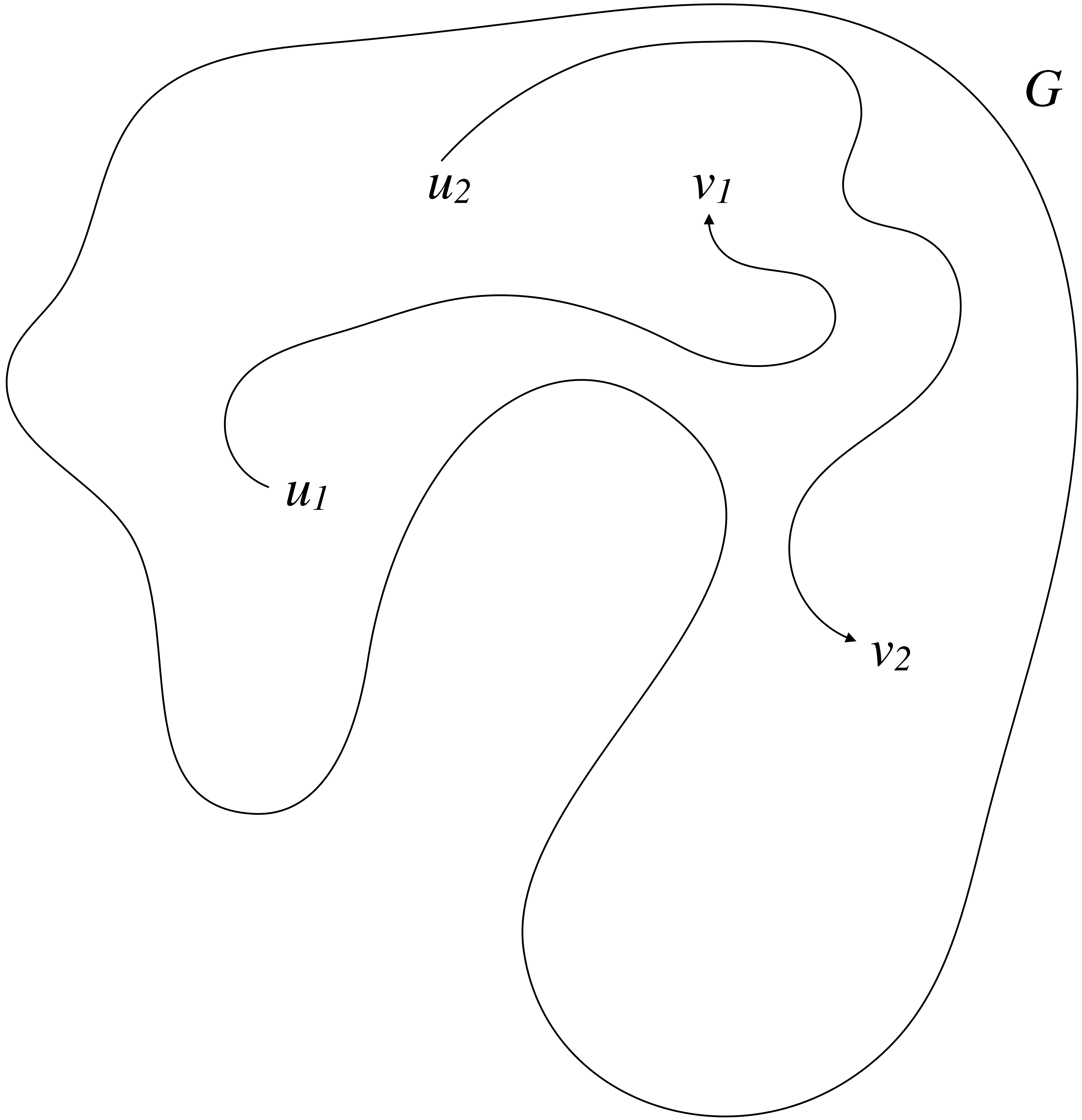
RPQs, Simple Path Semantics



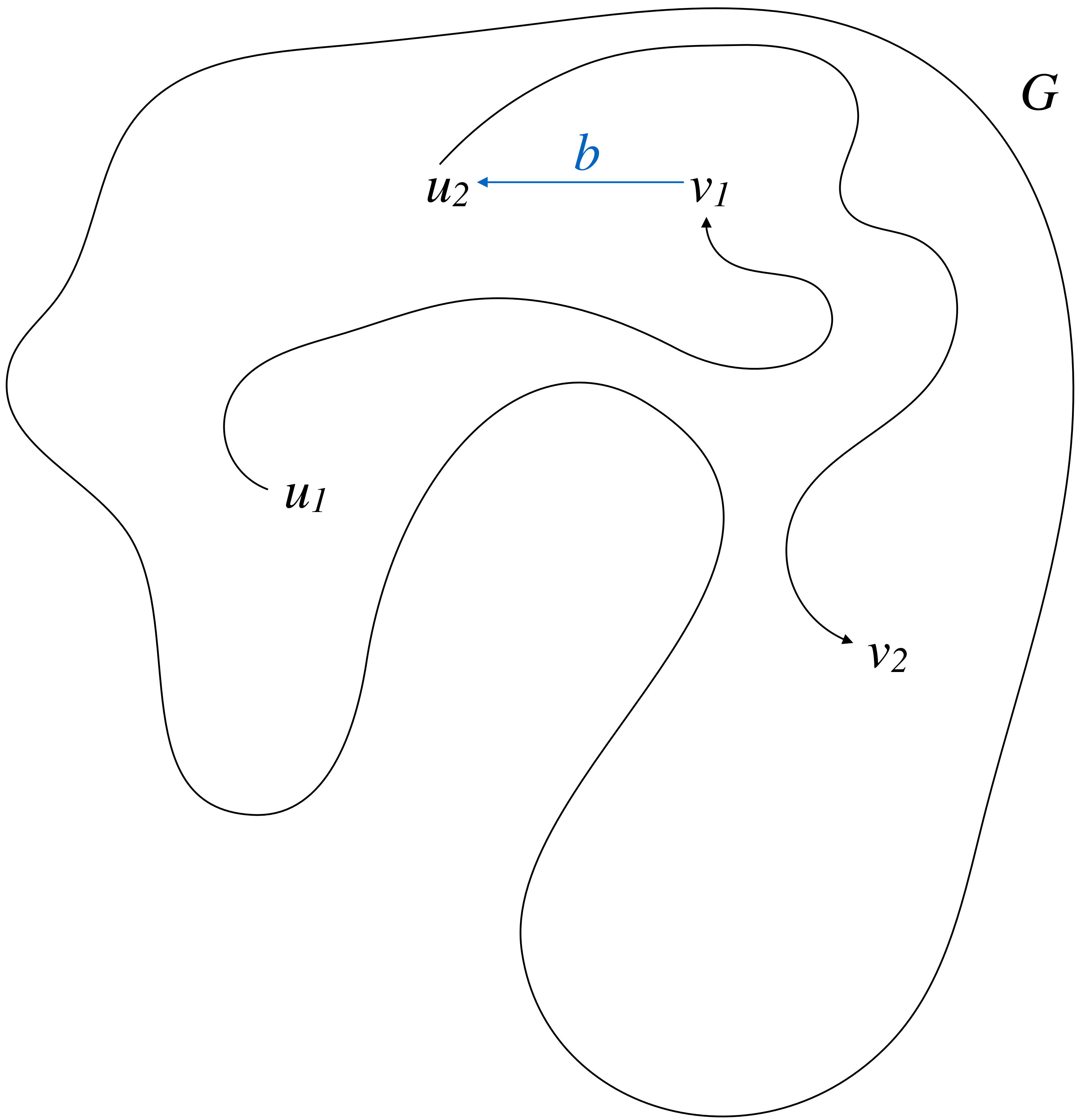
RPQs, Simple Path Semantics



RPQs, Simple Path Semantics



RPQs, Simple Path Semantics



RPQs, Trail Semantics

Theorem

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

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?

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

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

RPQs, Trail Semantics

Theorem

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

Reduction from

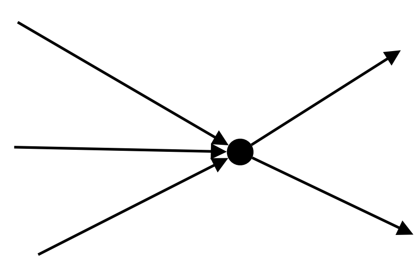
Two Edge Disjoint Paths

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



RPQs, Trail Semantics

Theorem

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

Reduction from

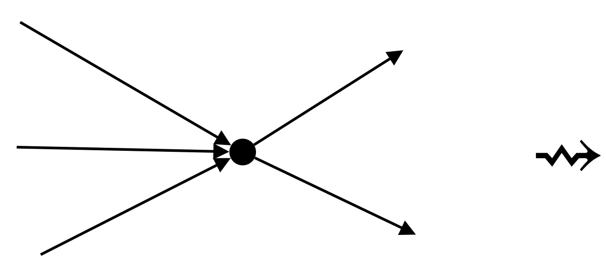
Two Edge Disjoint Paths

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



RPQs, Trail Semantics

Theorem

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

Reduction from

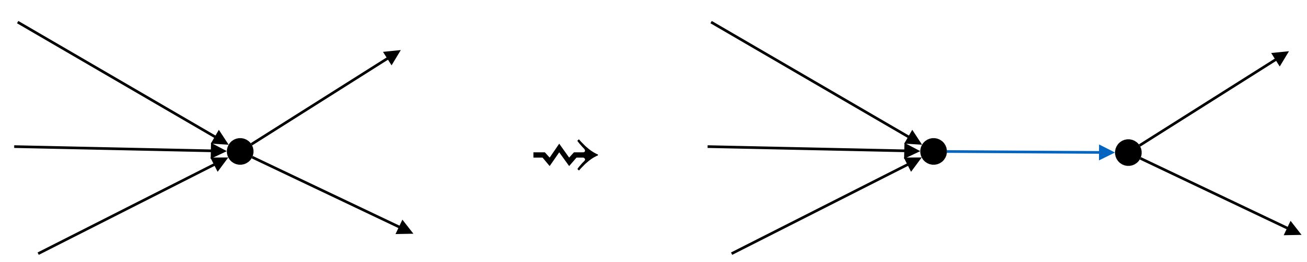
Two Edge Disjoint Paths

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



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]

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

$$(u_1, v_2) \in Q(G_b)_t$$

RPQs, Trail Semantics

Theorem

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

A similar proof.

CRPQs, Every Path Semantics

Theorem

CRPQ Evaluation under every path semantics is NP-complete

CRPQs, Every Path Semantics

Theorem

CRPQ Evaluation under every path semantics is NP-complete

Proof (sketch)

Lower bound: immediate from conjunctive queries

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 $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \land \cdots \land (y_n \xrightarrow{r_n} z_n)\big)$ be the query

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 $Q(\bar{x}) = \big((y_1 \overset{r_1}{\rightarrow} z_1) \wedge \cdots \wedge (y_n \overset{r_n}{\rightarrow} z_n)\big)$ be the query

For each regular expression r_i, we can compute in polynomial time

a relation R_i containing the pairs $r_i(G)$

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 $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be the query

For each regular expression r_i, we can compute in polynomial time a relation R_i containing the pairs $r_i(G)$

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

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 $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be the query

For each regular expression r_i, we can compute in polynomial time a relation R_i containing the pairs $r_i(G)$

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

$$Q_R(\bar{x}) = \big((y_1 \xrightarrow{R_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{R_n} z_n)\big)$$

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 $Q(\bar{x}) = \big((y_1 \xrightarrow{r_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{r_n} z_n)\big)$ be the query

For each regular expression r_i, we can compute in polynomial time a relation R_i containing the pairs $r_i(G)$

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

$$Q_R(\bar{x}) = \big((y_1 \xrightarrow{R_1} z_1) \wedge \cdots \wedge (y_n \xrightarrow{R_n} z_n)\big)$$

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 $C_{Rel} = \{Q_R \mid Q \in C\}$

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, 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

So, by the results on tree-shaped conjunctive queries,
evaluation on tree-shaped CRPQs is also tractable

CRPQs, Simple Path / Trail Semantics

Theorem

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

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

Overview

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

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

RPQ Containment

Theorem

RPQ Containment is PSPACE-complete

Theorem

CRPQ Containment is EXPSPACE-complete

RPQ Containment

Theorem

RPQ Containment is PSPACE-complete

Proof (sketch)

Theorem

CRPQ Containment is EXPSPACE-complete

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

Theorem

CRPQ Containment is EXPSPACE-complete

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 $L(r_1) \subseteq L(r_2)$

Theorem

CRPQ Containment is EXPSPACE-complete

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

Theorem

CRPQ Containment is EXPSPACE-complete

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

Theorem

CRPQ Containment is EXPSPACE-complete

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

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

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

$$x \sim y \qquad \text{or} \qquad x \nsim y$$

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

$$x \sim y \qquad \text{or} \qquad x \nsim y$$

satisfied if nodes x and y have the same, resp., different label (or data value)

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

$$x \sim y \qquad \text{or} \qquad x \nsim y$$

satisfied if nodes x and y have the same, resp., different label (or data value)

Such queries are usually considered on a different data model

(data words, data trees, data graphs)

but since we chose Σ 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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

$$\Sigma^* \cdot \downarrow x \cdot \Sigma^+[=x] \cdot \Sigma^*$$

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

$$\Sigma^* \cdot \downarrow x \cdot \Sigma^+[=x] \cdot \Sigma^*$$

expresses L_{eq}: bind x, see if it occurs elsewhere ($[=x]$)

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

$$\Sigma^* \cdot \downarrow x \cdot \Sigma^+[=x] \cdot \Sigma^*$$

expresses L_{eq}: bind x, see if it occurs elsewhere ($[=x]$)

Regular expressions with equality:

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

$$\Sigma^* \cdot \downarrow x \cdot \Sigma^+[=x] \cdot \Sigma^*$$

expresses L_{eq}: bind x, see if it occurs elsewhere ($[=x]$)

Regular expressions with equality:

$$\Sigma^* \cdot (\Sigma^+)_= \cdot \Sigma^*$$

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

Language L_{eq} is the most basic one imaginable that compares data values.
Hence regular expressions should avoid complementation.

Regular expressions with binding:

$$\Sigma^* \cdot \downarrow x \cdot \Sigma^+[=x] \cdot \Sigma^*$$

expresses L_{eq}: bind x, see if it occurs elsewhere ($[=x]$)

Regular expressions with equality:

$$\Sigma^* \cdot (\Sigma^+)_= \cdot \Sigma^*$$

also expresses L_{eq}: guesses where equal values occur