Run-based semantics for RPQs

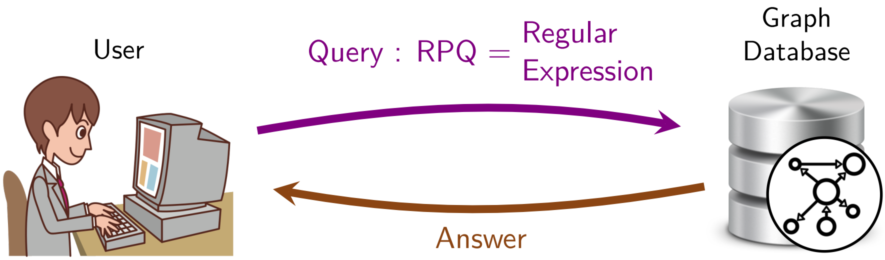
Victor MARSAULT*
joint work with Claire DAVID* and Nadime FRANCIS*

* Université Gustave-Eiffel, CNRS, LIGM

Verigraph Meeting, 2024-04-30, Grenoble, France

(mostly Claire slides used for KR'23)

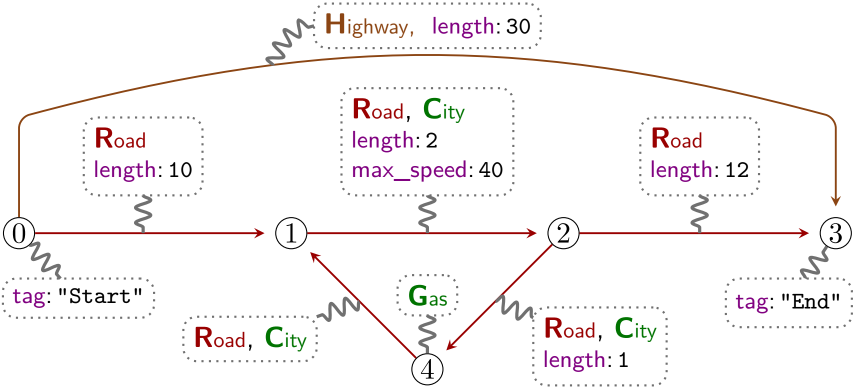
How to extract efficiently meaningful answers ?



What is a good semantics for RPQ's over graph DB's ?

- Meaningful answers
- Good complexity

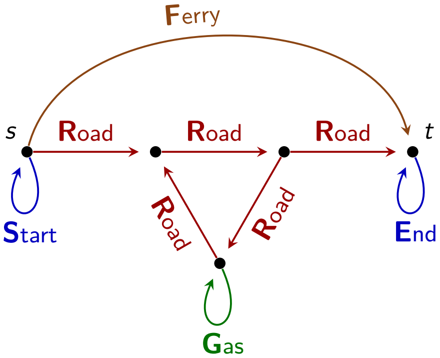
Graph DB - In practice Property graphs



RDF is another data model used in practice...

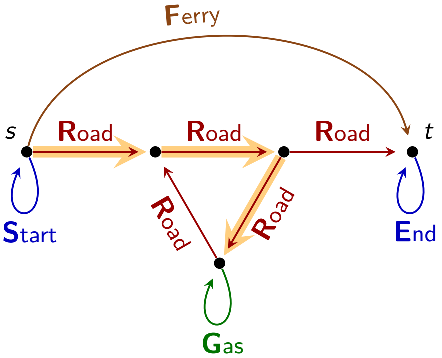
Graph DB - Our abstraction

- Finite label alphabet: $\Sigma = \{\mathbf{S}, \mathbf{R}, \mathbf{F}, \mathbf{G}, \mathbf{E}\}$
- Vertices
- Edges labelled over Σ



Graph DB - Our abstraction

- Finite label alphabet:
  $\Sigma = \{\mathbf{S}, \mathbf{R}, \mathbf{F}, \mathbf{G}, \mathbf{E}\}$
- Vertices
- Edges labelled over Σ



Terminogy: Walk

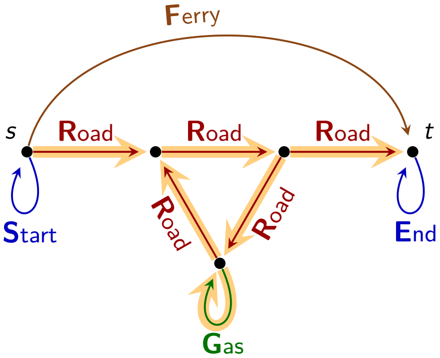
- Consistent sequence of edges
- Repetitions are allowed

Graph DB - Our abstraction

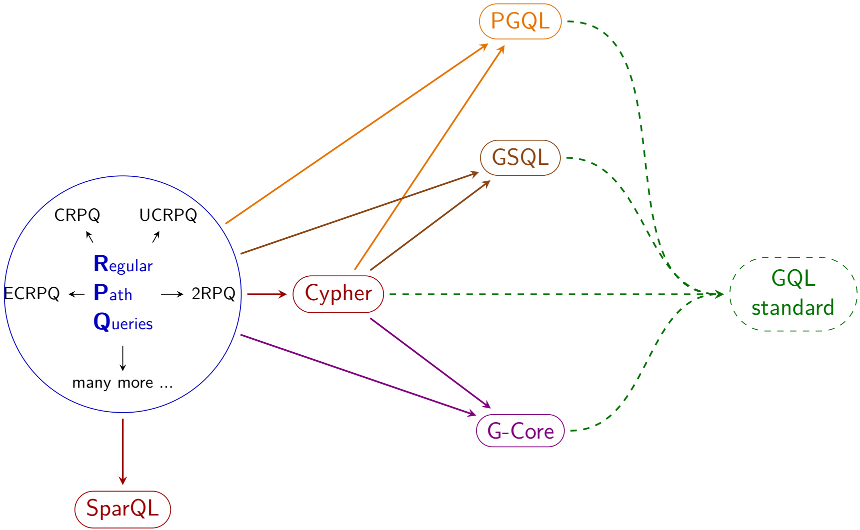
- Finite label alphabet: $\Sigma = \{\mathbf{S}, \mathbf{R}, \mathbf{F}, \mathbf{G}, \mathbf{E}\}$
- Vertices
- Edges labelled over Σ

Terminogy: Walk

- Consistent sequence of edges
- Repetitions are allowed



RPQs - Core of most query languages for graph DB



RPQ - in theory

$$
\begin{array}{lll}
Q ::= \mathbf{A} & & \text{Atoms} \\
& QQ & \text{Concatenation} \\
& Q+Q & \text{Disjunction} \\
& Q^* & \text{Kleene star}
\end{array}
$$

where $\mathbf{A}$ is a label in the graph.

Label of a walk

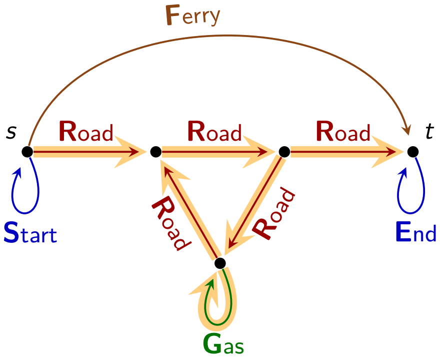
Contenation of labels of edges

Ex : **RRRGRRR**

Definition: match for Q

A walk w such that the label of w matches Q.

Ex: match for $\mathbf{R}^*\mathbf{G}\mathbf{R}^*$

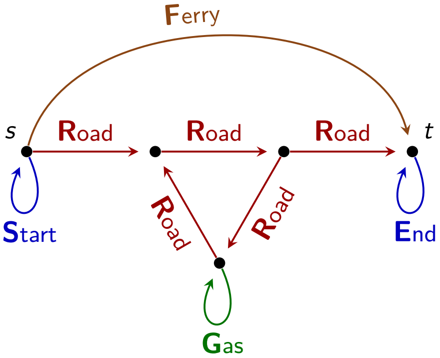


Our two running RPQs

"Find a way from s to t"

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_1?

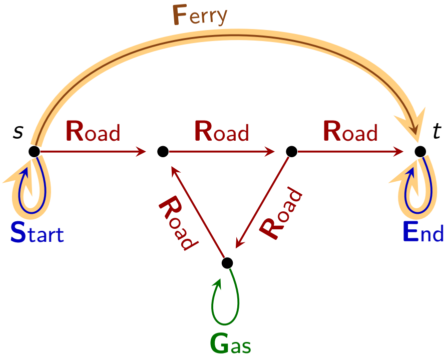


"Find a way from s to t"

$Q_1 = \;\; \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_1?

- The ferry

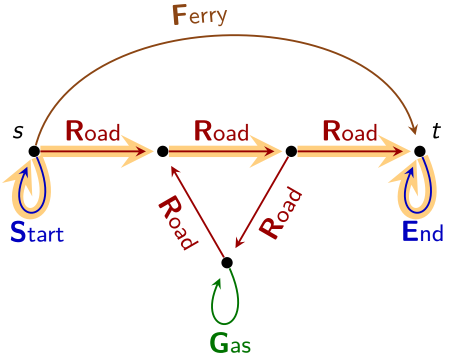


"Find a way from s to t"

$Q_1 = \; \mathbf{S}\;(\mathbf{R}+\mathbf{F})^*\;\mathbf{E}$

Which walks match Q_1?

- The ferry
- The straight road

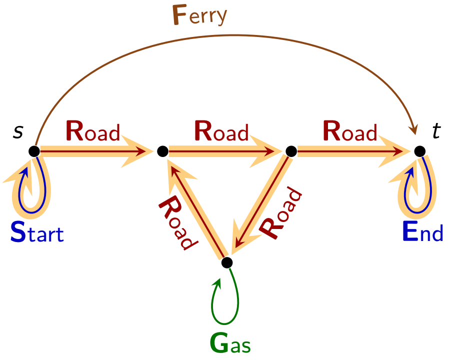


"Find a way from s to t"

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_1?

- The ferry
- The straight road
- Walks with some circuit laps



Our two running RPQs

"Find a way from s to t"

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

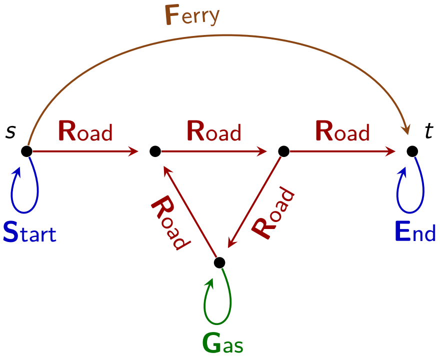
Which walks match Q_1?

- The ferry
- The straight road
- Walks with some circuit laps

"...with mandatory gas stop"

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_2?



"Find a way from s to t"

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_1?

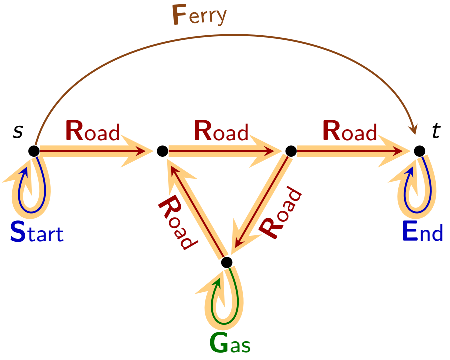
- The ferry
- The straight road
- Walks with some circuit laps

"...with mandatory gas stop"

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_2?

- Walks with some circuit laps



"Find a way from s to t"

$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

Which walks match Q_1?

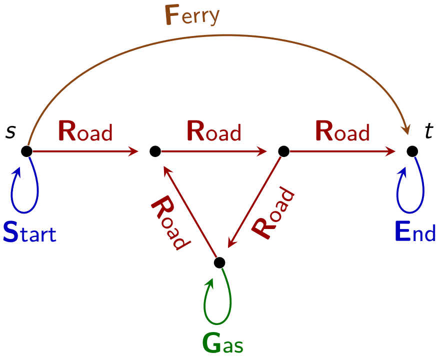
- The ferry
- The straight road
- Walks with some circuit laps

"...with mandatory gas stop"

$Q_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

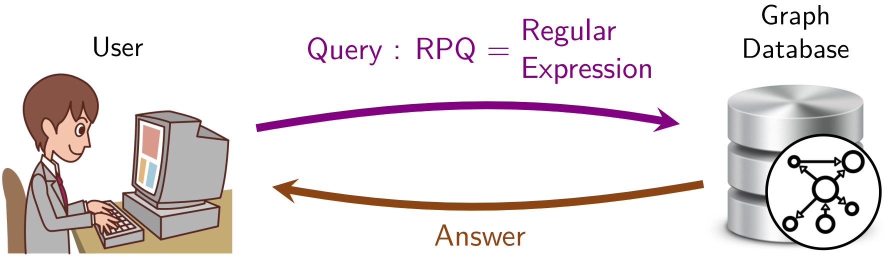
Which walks match Q_2?

- Walks with some circuit laps



$\Rightarrow$ Infinitely many matches

How to extract efficiently meaningful answers ?



What is a good semantics for RPQ's over graph DB's ?

- Meaningful answer
- Good complexity

⚠️ Infinitely many matches but users expect a finite answer ⚠️

Homomorphism semantics

Main theoretical semantics, SPARQL (RDF)

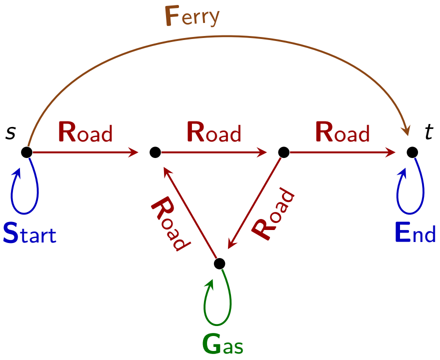
Definition

- Returns the endpoints of matching walks

$$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

$$Q_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

- Both return one pair: (s, t)

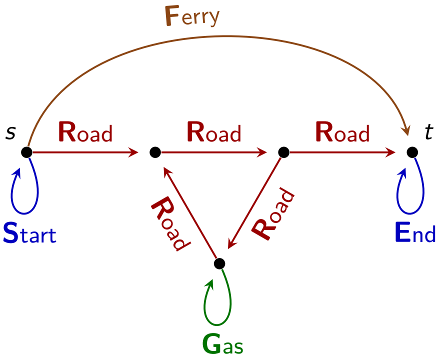


Homomorphism semantics

Main theoretical semantics, SPARQL (RDF)

Definition

- Returns the endpoints of matching walks



$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

$Q_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Both return one pair: (s, t)

- Well grounded theory 🥳
- Efficient algorithms 😀
- Returns little information ☹️

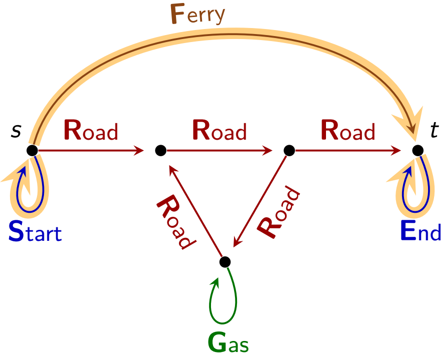
PGQL (Oracle), GSQL (TigerGraph), G-Core [Angles et al. 2018], GQL

Definition

- Returns the walk with the least number of edges

$Q_1 = $ **S** (**R**+**F**)* **E**

- Returns the ferry
- Walks using roads have more edges



Shortest-walk semantics

PGQL (Oracle), GSQL (TigerGraph), G-Core [Angles et al. 2018], GQL

Definition

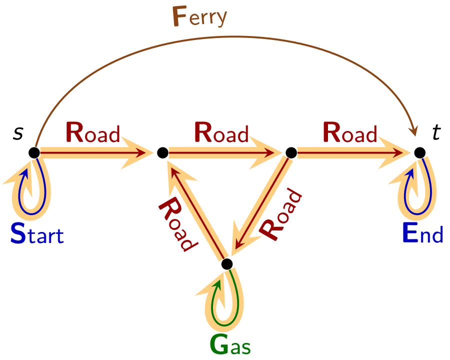
- Returns the walk with the least number of edges

$Q_1 = \; \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns the ferry
- Walks using roads have more edges

$Q_2 = \; \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns the walk with one circuit lap



Shortest-walk semantics

PGQL (Oracle), GSQL (TigerGraph), G-Core [Angles et al. 2018], GQL

Definition

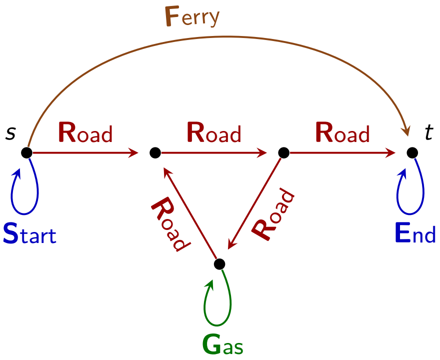
- Returns the walk with the least number of edges

$$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

- Returns the ferry
- Walks using roads have more edges

$$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

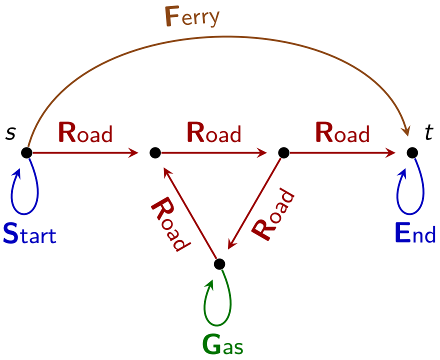
- Returns the walk with one circuit lap



- Efficient algorithms 😀
- Arbitrary choice of witness ☹️
- No vertical post-processing ☹️

Definition

- Returns walks
- Each edge can be used at most once



Trail semantics

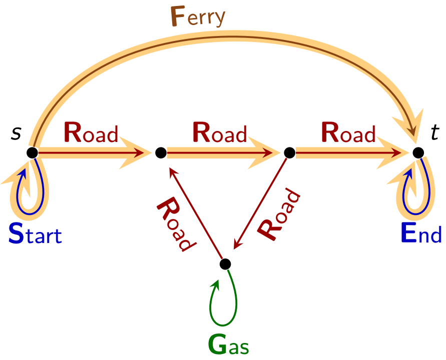
Cypher (Neo4j), GQL

Definition

- Returns walks
- Each edge can be used at most once

$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_1 returns
  - the ferry
  - the straight road



Trail semantics

Cypher (Neo4j), GQL

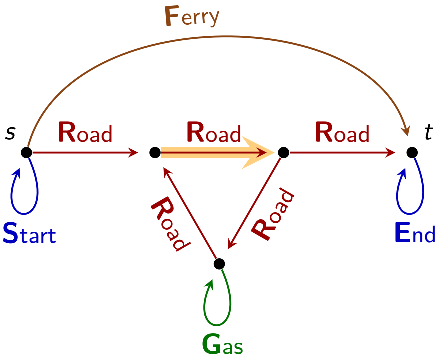
Definition

- Returns walks
- Each edge can be used at most once

$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_1 returns
  - the ferry
  - the straight road
- Walks with circuit laps

  $\Rightarrow$ repeat the middle edge



Trail semantics

Cypher (Neo4j), GQL

Definition

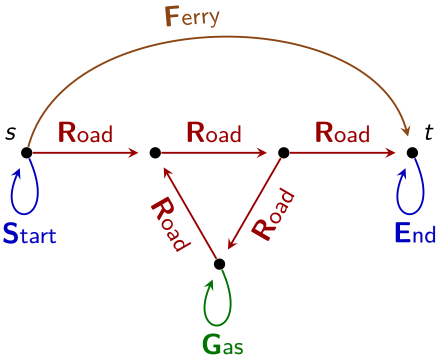
- Returns walks
- Each edge can be used at most once

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_1 returns
  - the ferry
  - the straight road
- Walks with circuit laps
  $\Rightarrow$ repeat the middle edge

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_2 returns no results !!



Trail semantics

Cypher (Neo4j), GQL

Definition

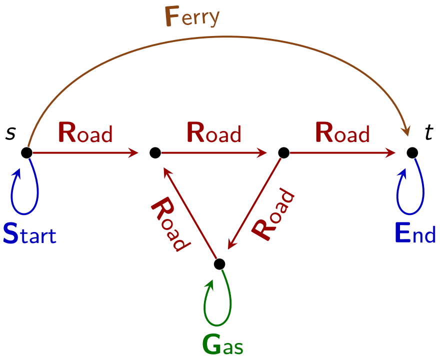
- Returns walks
- Each edge can be used at most once

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_1 returns
  - the ferry
  - the straight road
- Walks with circuit laps
  $\Rightarrow$ repeat the middle edge

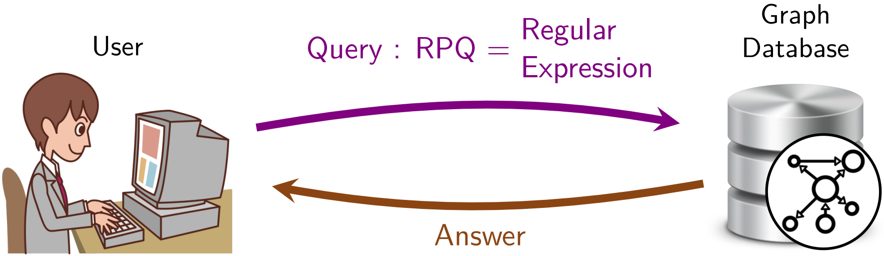
$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Q_2 returns no results !!



- Enables post-processing 🥳
- Problems are untractable ☹️
- Discards meaningful matching walks ☹️

What is a good semantics for RPQ's over graph DB's?



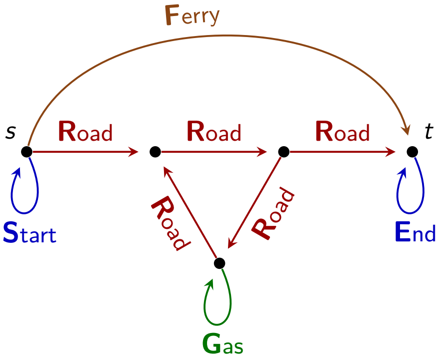
Infinitely many matches but the user expects a finite answer

- Homomorphism → Filters out most information
- Shortest-walk → Bad "coverage" of matching walks
- Trail → Problems are computationally hard
  May discard meaningful matching walks

No solution is clearly superior

Definition

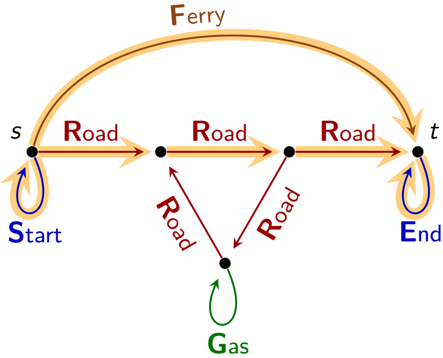
- Returns walks
- Each edge can match each atom of Q at most once



Our proposal: binding-trail semantics (1)

Definition

- Returns walks
- Each edge can match each atom of Q at most once



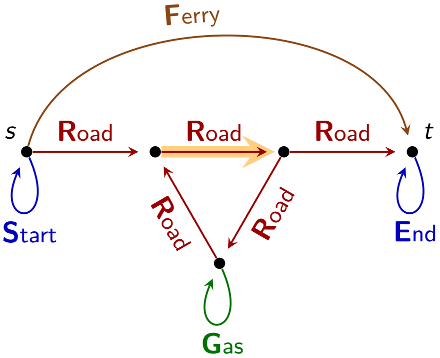
$Q_1 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns
  - the ferry
  - the straight road

Our proposal: binding-trail semantics (1)

Definition

- Returns walks
- Each edge can match each atom of Q at most once



$Q_1 = \; \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

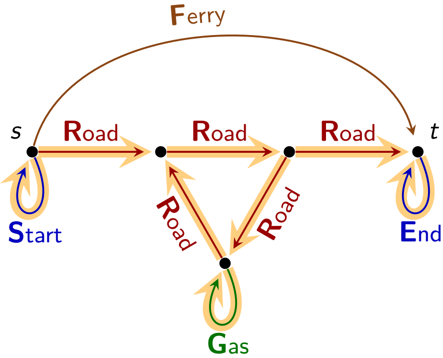
- Returns
  - the ferry
  - the straight road
- In walks with circuit laps $\Longrightarrow$ the middle edge reuses **R**

Definition

- Returns walks
- Each edge may use once each atom in Q

$$Q_2 = \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

- Returns the walk with one circuit lap



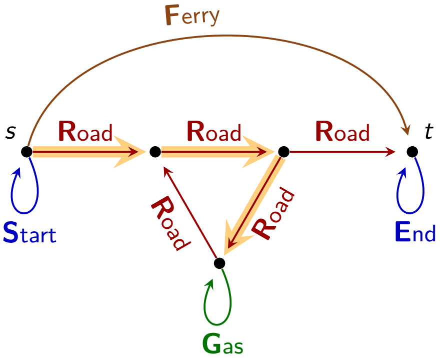
Our proposal: binding-trail semantics (2)

Definition

- Returns walks
- Each edge may use once each atom in Q

$$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

- Returns the walk with one circuit lap
  - Before **G** $\rightarrow$ use the left **R**

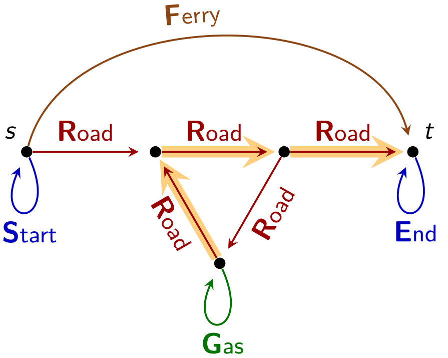


Definition

- Returns walks
- Each edge may use once each atom in Q

$Q_2 = \ \ \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- Returns the walk with one circuit lap
  - Before **G** $\rightarrow$ use the left **R**
  - After **G** $\rightarrow$ use the right **R**



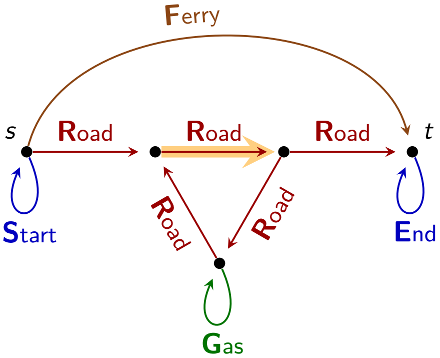
Our proposal: binding-trail semantics (2)

Definition

- Returns walks
- Each edge may use once each atom in Q

$$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$$

- Returns the walk with one circuit lap
  - Before **G** $\rightarrow$ use the left **R**
  - After **G** $\rightarrow$ use the right **R**
- In walks with 2+ circuit laps
  $\Longrightarrow$ the middle edge reuses the left **R** or the right **R**

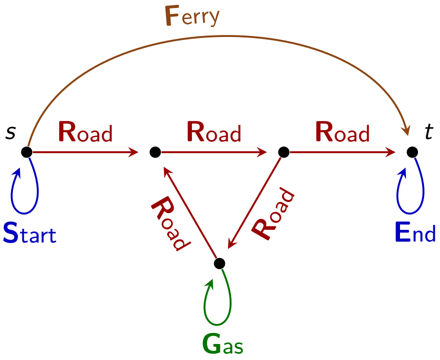


$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- The ferry
- The straight road

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- The walk with one circuit lap



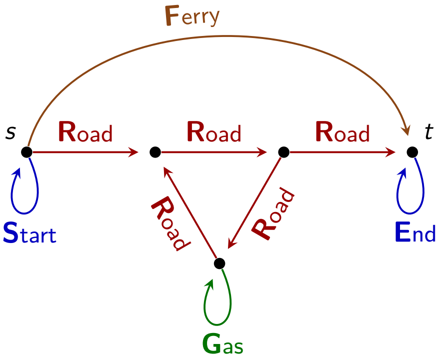
Binding-trail semantics provides "better coverage" 🥳

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

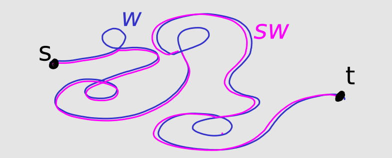
- The ferry
- The straight road

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- The walk with one circuit lap



Lemma



$\forall$ match w of Q
$\implies$ some subwalk sw returned

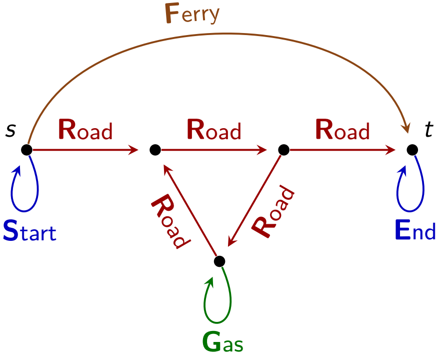
Binding-trail semantics provides "better coverage" 🥳

$Q_1 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

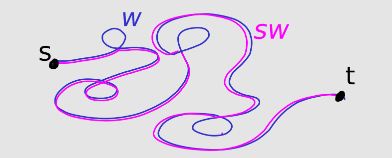
- The ferry
- The straight road

$Q_2 = \mathbf{S}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{G}\ (\mathbf{R}+\mathbf{F})^*\ \mathbf{E}$

- The walk with one circuit lap



Lemma



$\forall$ match w of Q

$\implies$ some subwalk sw returned

Open questions 🧐

- Define "coverage"
- Define "good" coverage

Retro-compatibilities

Binding-trail is compatible with Homomorphism

Homomorphism semantics returns (s, t)
$\iff$ Binding-trail semantics returns some walk w from s to t.

$\implies$ If users need endpoints only, use algorithm for homomorphism

Binding-trail is compatible with Shortest-walk

Shortest matching walks $\subseteq$ Binding-trail matching walks.

$\implies$ If users need only one walk, use algorithm for shortest-walk

Binding-trail is compatible with Trail

Matching trails $\subseteq$ Binding-trail matching walks.

Computational properties of binding-trail semantics

Tractable problems 😀

1. **Emptiness** is NL-complete.
2. **Enumerating** the **bag** of answers is Poly-delay.

Untractable problems ☹️

3. **Counting** the number of matched walk is $\#P$ complete.
4. **Membership** of a given walk is NP-complete.

About 3 : Counting is $\#P$ complete for any reasonable semantics.
About 4 : Mostly a theoretical concern.

Open problem 🧐

5. **Enumerating** the **set** of answers.

The output depends on the syntax of the query 🤔

$\mathbf{R}^*$

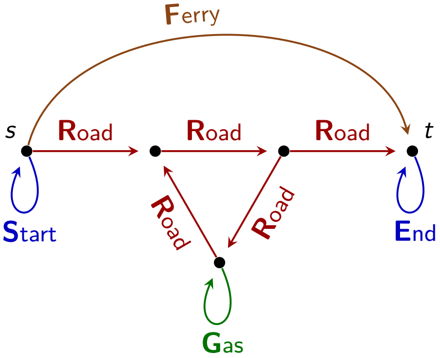
- allows no lap in the circuit

$\mathbf{R}^*\mathbf{R}^*$

- allows 1 lap in the circuit

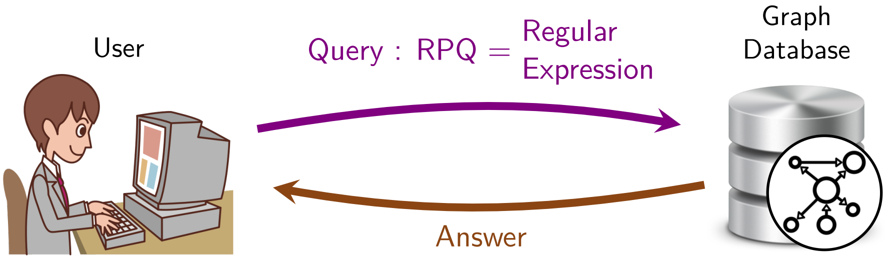
$(\mathbf{R} + \mathbf{R})^*$

- allows 1 lap in the circuit
- In general, $\neq \mathbf{R}^*\mathbf{R}^*$



Unusual from theoretical point of view 🤯
The user has finer control on the output ☺️
This kind of syntax quirks exists in practice 😵‍💫

Run-based semantics, a reasonable compromise?



- 🥳 Coverage of matches
- 🥳 Tractable emptiness and enumeration
- 🤔 Syntax-dependent

Perspectives 🧐

- Deduplicated enumeration
- Containment
- How to deal with data values?
- Get binding-trail into GQL 🤩

How to decide if a semantics is good?

- Theoreticians mostly worry about complexity 🧑‍🎓
- But complexity is not the only relevant criterion
  Why? Trail semantics is the most popular in industry 👷

Ideas for a comparison framework 🏗️

- Coverage (horizontal aggregation)
- Monotonicity (distributed databases)
- Compatibility with operators (result predictability)
- Kinds of definition (selector/restrictor in GQL)

Promising semantics 🤞

- Undominated semantics (minimal for the subwalk order)
- Shortest coverage semantics

Appendix

Comparison of semantics

Semantics	Shortest-walk	Trail	Run-based
Existence	■ Tractable	■ Untractable	■ Tractable
Enumeration	■ Tractable	■ Untractable	■ Tractable
Distinct Enum	■ Tractable	■ Untractable	Open
Counting	■ Meaningless	■ Untractable	■ Untractable
Walk Memb.	■ Tractable	■ Tractable	■ Untractable
Coverage	■ None	■ No guarantee	■ “Subwalk” guarantee

Tools and technics

Simple run

- Query given as an automaton
- Outputs simple walks in the product $D \times \mathcal{A}_Q$
- Good formal setting for theory

Binding trail

- Query given as a RegExp
- Outputs matching walks that do not reuse edge on a same atom
- Closer to practice

Theorem

The two semantics are computationally equivalent.

Key idea for $\Longrightarrow$

From an automaton $\mathcal{A}$, we build a regular expression E such that runs of $\mathcal{A}$ are encoded into runs of the Glushkov automaton of E.