EDBT ICDT 2026 TAMPERE

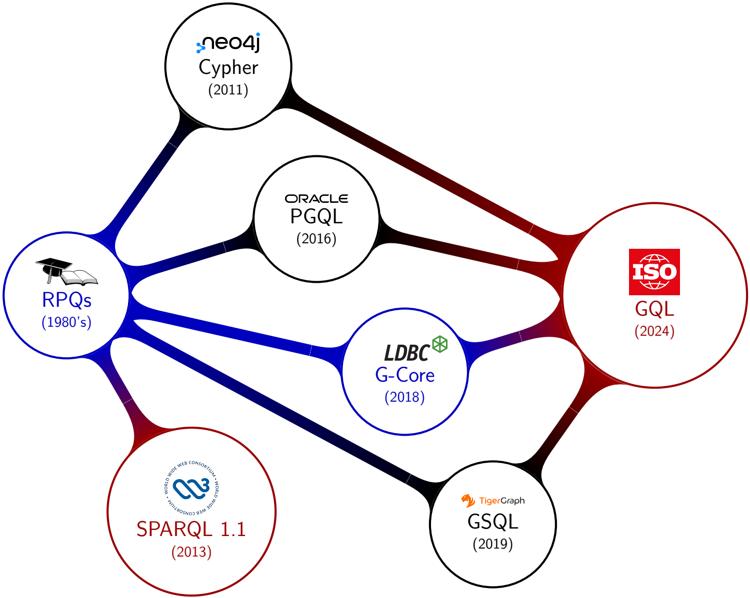
March 24

Designing and Comparing RPQ Semantics

Victor Marsault Antoine Meyer

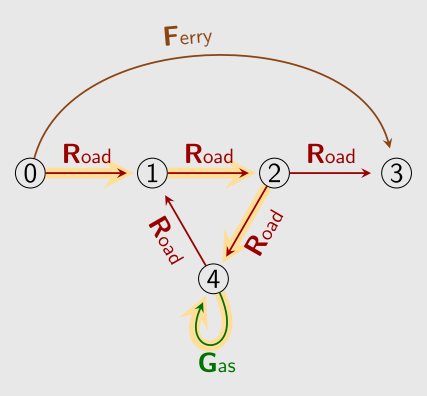
Univ. Gustave Eiffel, CNRS, LIGM – France

Overview of graph query languages



Data model and query language

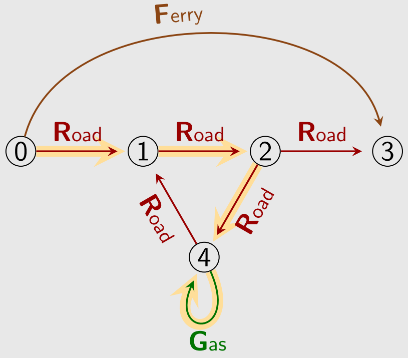
Labeled graph D



Nodes, Edges, Labels, Paths

Data model and query language

Labeled graph D



Nodes, Edges, Labels, Paths

Regular Path Query Q

- Input: a regexp over labels

  Ex: $(\mathbf{R} + \mathbf{F})^* \mathbf{G}$

- Match: path with a label that conforms to Q

  Ex: $0 \xrightarrow{\mathbf{R}} 1 \xrightarrow{\mathbf{R}} 2 \xrightarrow{\mathbf{R}} 4 \xrightarrow{\mathbf{G}} 4$

$\textsc{Matches}(D, Q)$: match set

- Issue: $\textsc{Matches}(D, Q)$ is sometimes infinite

Three ways to keep the output finite

Endpoint semantics

- Outputs the **endpoints** of matches

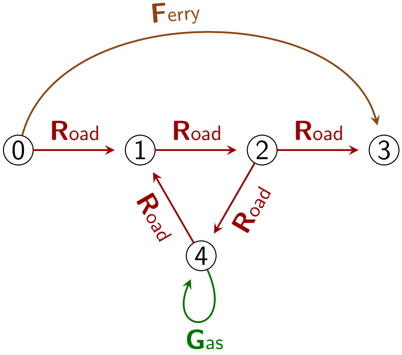
Trail semantics

- Outputs **matches** with **no repeated edges**

Shortest semantics

- Outputs **matches** with the **minimum number of edges**

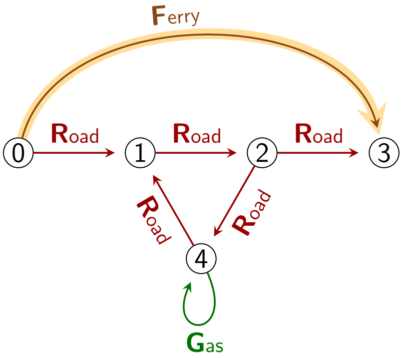
Example: Find paths from 0 to 3



Three kinds of $0 \rightsquigarrow 3$ paths:

- The ferry
- The straight road
- Some road path with laps

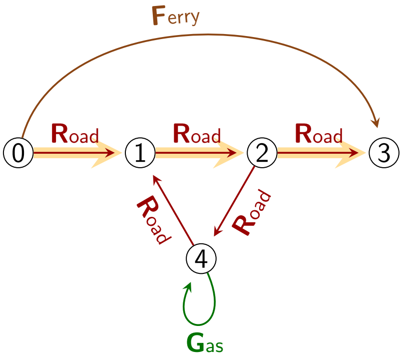
Example: Find paths from 0 to 3



Three kinds of 0 ⇝ 3 paths:

- The ferry
- The straight road
- Some road path with laps

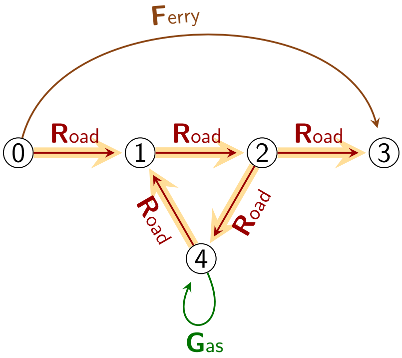
Example: Find paths from 0 to 3



Three kinds of $0 \rightsquigarrow 3$ paths:

- The ferry
- The straight road
- Some road path with laps

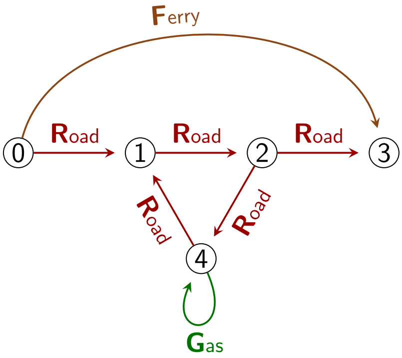
Example: Find paths from 0 to 3



Three kinds of $0 \rightsquigarrow 3$ paths:

- The ferry
- The straight road
- Some road path with laps

Example: Find paths from 0 to 3



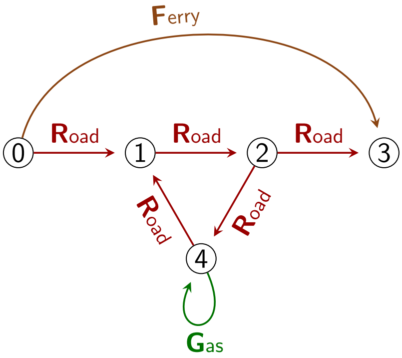
$$Q_1 = (\mathbf{R} + \mathbf{F})^*$$

- Endpoint outputs $(0, 3)$
- Shortest outputs the ferry
- Trail outputs the ferry and the straight road

Three kinds of $0 \rightsquigarrow 3$ paths:

- The ferry
- The straight road
- Some road path with laps

Example: Find paths from 0 to 3



Three kinds of $0 \rightsquigarrow 3$ paths:

- The ferry
- The straight road
- Some road path with laps

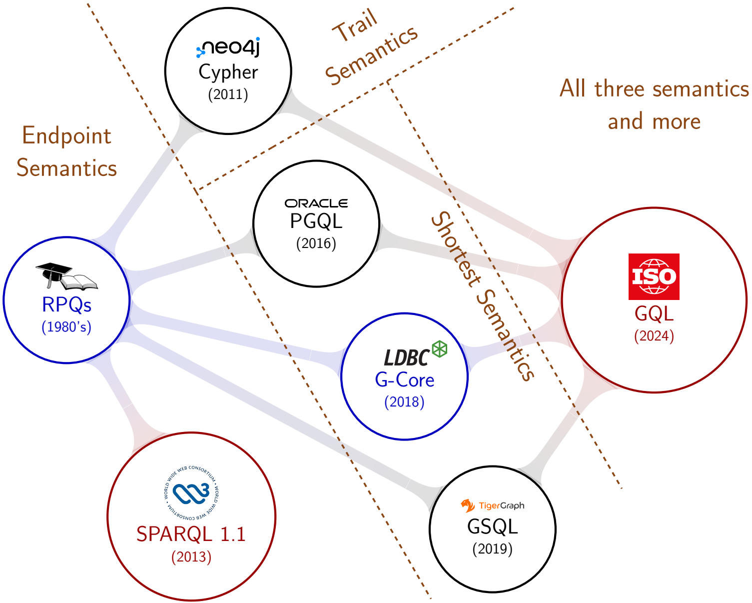
$$Q_1 = (\mathbf{R} + \mathbf{F})^*$$

- Endpoint outputs $(0, 3)$
- Shortest outputs the ferry
- Trail outputs the ferry and the straight road

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

- Endpoint outputs $(0, 3)$
- Shortest outputs the road with one lap
- Trail has no output

Overview of graph query languages



Trail vs shortest

Complexity perspective

	Shortest	Trail
Existence *"Is there a path in the output?"*	■ Tractable	■ Untractable
Enumeration *"Enumerate all paths in the output"*	■ Tractable	■ Untractable
Membership *"Is a given path in the output?"*	■ Tractable	■ Tractable

Trail vs shortest

Complexity perspective

	Shortest	Trail
Existence *"Is there a path in the output?"*	■ Tractable	■ Untractable
Enumeration *"Enumerate all paths in the output"*	■ Tractable	■ Untractable
Membership *"Is a given path in the output?"*	■ Tractable	■ Tractable

Questions

- Why was trail (Neo4j/Cypher) the most successful?
- What makes a semantics good?
- Can we design better semantics?

In the abstract, what is an RPQ semantics?

RPQ semantics = function S

$$S\colon (D, Q) \longmapsto O \quad \subseteq_{\mathsf{fin}} \textsc{Matches}(D, Q)$$

- Database D
- RPQ Q
- **Finite** set O of outputs

Ex: Trail and Shortest

In the abstract, what is an RPQ semantics?

RPQ semantics = function S

$$S\colon (D, Q) \longmapsto O \quad \subseteq_{\mathsf{fin}} \textsc{Matches}(D, Q)$$

- Database D
- RPQ Q
- **Finite** set O of outputs

Ex: Trail and Shortest

Filter-based semantics

*"Keeps the **good** matches w.r.t. some boolean function f"*

$f(p) = \text{good} \iff p \text{ is a trail}$

In the abstract, what is an RPQ semantics?

RPQ semantics = function S

$$S\colon (D, Q) \longmapsto O \quad \subseteq_{\text{fin}} \textsc{Matches}(D, Q)$$

- Database D
- RPQ Q
- **Finite** set O of outputs

Ex: Trail and Shortest

Filter-based semantics

"Keeps the **good** *matches w.r.t. some boolean function f"*

$f(p) = \text{good} \iff p$ is a trail

Order-based semantics

"Keeps the **best** *matches w.r.t. some well-quasi-order $\prec$"*

$p_1 \prec p_2 \iff p_1$ is shorter than p_2

Some desirable properties

Given $\quad S\colon (D, Q) \longmapsto O \quad \subseteq_{\mathsf{fin}} \textsc{Matches}(D, Q)$

Some desirable properties

Given $S\colon (D, Q) \longmapsto O \quad \subseteq_{\text{fin}} \textsc{Matches}(D, Q)$

S is compatible with $+$

$$S(D, Q_1 + Q_2) = S(D, Q_1) \cup S(D, Q_2)$$

S is compatible with $\cdot$

...

S is compatible with $*$

...

$\Rightarrow$ Bottom-up computation
$\Rightarrow$ User Explainability

Some desirable properties

Given $S\colon (D, Q) \longmapsto O \quad \subseteq_{\text{fin}} \textsc{Matches}(D, Q)$

S is compatible with $+$

$$S(D, Q_1{+}Q_2) = S(D, Q_1) \cup S(D, Q_2)$$

S is compatible with $\cdot$

...

S is compatible with $*$

...

⇒ Bottom-up computation
⇒ User Explainability

S is monotonous

$$D_1 \subseteq D_2 \Rightarrow S(D_1, Q) \subseteq S(D_2, Q)$$

⇒ Distributed databases

Some statements

Under various mild hypotheses,

- A semantics is never $\cdot$-compatible, nor $*$-compatible
- Compatible with $+$ $\iff$ Filter-based

Some statements

Under various mild hypotheses,

- A semantics is never $\cdot$-compatible, nor $*$-compatible
- Compatible with $+ \iff$ Filter-based
- Filter-based semantics are monotonous
- Order-based semantics are **not** monotonous
  (Adding new elements might create a shortcut)

Trail vs shortest

Complexity & properties

	Shortest	Trail
Existence *"Is there a path in the output?"*	■ Tractable	■ Untractable
Enumeration *"Enumerate all paths in the output"*	■ Tractable	■ Untractable
Membership *"Is a given path in the output?"*	■ Tractable	■ Tractable
Monotonous	■ No	■ Yes
Compatible with $+$	■ No	■ Yes
Compatible with $\cdot / *$	■ No	■ No

Example of a new semantics

Example of a new semantics

Subpath Minimal semantics

- Order-based semantics
- $p_1 \preceq p_2 \iff p_1$ is a subsequence of p_2

Example of a new semantics

Subpath Minimal semantics

- Order-based semantics
- $p_1 \preceq p_2 \quad \Longleftrightarrow \quad p_1$ is a subsequence of p_2

Properties

- It is monotonous
  (unlike most order-based semantics)
- It guarantees some "coverage" of $\textsc{Matches}(D, Q)$
  (only removes cycles, and only if they are unnecessary)

Conclusion

- Practical RPQ semantics keep a finite number of matches
- Few ad-hoc semantics were used in practice
- ...and then query evaluation efficiency was studied

Systematic study of RPQ semantics as functions

- Classes and properties
  $\Rightarrow$ Compare existing semantics
- Relations between them (implications, incompatibilies, etc.)
  $\Rightarrow$ Design of future RPQ semantics

Future work

- Many ways to extend the framework
- New interesting semantics to study