Querying graphs with regular expressions

Abstract

Graph database management systems have increased in popularity over the last decades. In database theory, we abstract such databases as labelled graphs. Most real query languages are based on the well-known formalism of regular path queries (RPQs). Such a query is defined by a regular expression R. Any walk in the graph labelled with a word conforming to R is called a match, and in general there are infinitely many matches. The main challenge is to efficiently compute a finite and meaningful output from the matches.

Several approaches are used in practice and theory to reach this goal. Homomorphism semantics is the most studied and enjoy nice theoretical properties, but is not suitable for some practical applications (too little information is kept in the output). On the other side of the spectrum, the most widespread semantics in practice is called trail semantics and seems unreasonable from a theoretical standpoint (high complexity, arbitrary restrictions).

In a recent work, we suggested a new approach, run-based semantics, which seems to be a reasonable compromise. It restricts the infinitely many matches to a finite number by stopping when a cycle occurs in the computation of the query and in the graph simultaneously. The internship is about exploring RPQ semantics, their properties and connections.

1 Practical details

- Advisor: Victor Marsault
- Co-advisors: subset of {Claire David, Nadime Francis, Antoine Meyer}
- Laboratory: LIGM (Laboratoire d'Information Gaspard Monge)
- Team: BAAM (Bases de données, Automates, Analyse d'algorithmes et Modèles)
- Location: Université Gustave-Eiffel (RER A *Noisy-Champs*, 30 minutes from Paris)

2 Example of graph databases, RPQs and matches

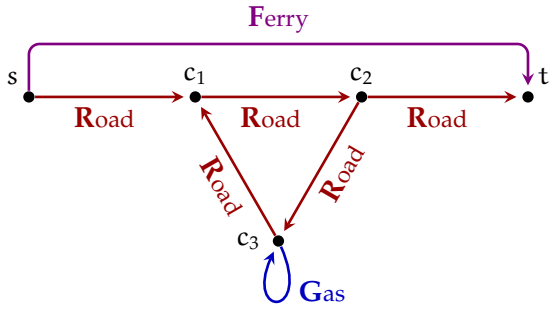

Figure 1: A graph database D

$R_1 = (\mathbf{R} + \mathbf{F})^*$

$R_2 = (\mathbf{R} + \mathbf{F})^* \, \mathbf{G} \, (\mathbf{R} + \mathbf{F})^*$

Figure 2: R_1, a simple reachability RPQ, and R_2, reachability with a mandatory stop

Here are two walks and whether they are matches to R_1, R_2 or both.

$s \to c_1 \to c_2 \to t$	match to R_1 but not R_2
$s \to c_1 \to c_2 \to c_3 \to c_3 \to c_1 \to c_2 \to t$	match to R_1 and R_2

3 Description

Graph databases have increased in popularity over the last decades. In database theory, we abstract such databases as labelled graphs (see Fig. 1). Most real query languages are based on Regular Path Queries (RPQs): an RPQ R is defined by a regular expression R (see Fig. 2). A *match* to R is any walk (that is, any sequence of edges) labelled by a word that conforms to R. For instance, $w_1 = s \to c_1 \to c_2 \to t$ is labelled by **RRR**, hence is a match to R_1 but not to R_2, and $w_2 = s \to c_1 \to c_2 \to c_3 \to c_3 \to c_1 \to c_2 \to t$ is a match to both R_1 and R_2. RPQs are traditionally evaluated under *homomorphism semantics*, which means that they return only the endpoints of matches: the pair (s, t) is returnd by both R_1 and R_2 because they both match w_2.

Homomorphism semantics enjoys nice theoretical properties but does not fit all applications. Indeed, one might need the number of matches (Tuple Multiplicity[1]), or even to enumerate matches for further processing (Walk Enumeration). One would want to returns matches directly, but since there are infinitely many of them, the different semantics use different filters to return only finitely many matches. *Trail semantics* discards all walks that repeated edge, which makes most problems at least NP-hard. *Shortest-walk semantics* keeps only walks with a minimal number of edges. Most computational problems are in P-TIME (or equivalent) but some problems are meaningless (Tuple Multiplicity[1]). Issues also arise from the fact that the metrics is arbitrary: R_1 returns the ferry route $s \to t$ over the straight road $s \to c_1 \to c_2 \to t$ but it is arguably less relevant.

Recently, we proposed [DFM23] new semantics, called binding-trail, that seems to be a good compromise. Similarly to trail semantics, it discards cyclic results, but only if a cycle in the walk coincides with a cycle in the computation of the query. Some important computational problems are PTime (Tuple Membership[1], Walk Enumeration), others remain NP-hard (Tuple Multiplicity, Walk Membership). Binding-trail semantics seems to have better behavior with respect to other unformal criteria that remain to be formalised.

The internship is about further investigating the different semantics of RPQs and their properties. Here are a few examples of research direction.

Run-based semantics remains to be explored for the most part. For instance, Distinct Walk Enumeration[1]is an important problem in practice, and its complexity is still open under run-based semantics. More applied directions are also possible. For instance, we think run-based semantics could be adapted to be used in practice, namely in the language GQL [Deu+22], the first standard language for querying property graphs.

Studying the complexity of computational problems is not enough to compare the semantics of RPQs. It misses the fact that a semantics must return sensible results. For instance, it seems that trail semantics does a better job than shortest-walk semantics to cover all "possibilities" offered by matches. For instance, under trail semantics, R_1 returns both the ferry route $s \to t$ and the direct road $s \to c_1 \to c_2 \to t$, which are the two reasonable possibilities (other matching walks take unnecessary turns in the circuits). This notion of *coverage* remains to be defined and studied. It would be part of developing a framework to compare and classify RPQs semantics.

4 Prior knowledge

The student is expected to know the basics of formal methods (complexity, formal languages, logic).

[1]Computational problems are briefly described in Section 5.

5 Short description of the computational problems

Note that all problems below are parameterised by the some semantics S.

Tuple Membership – Given a graph database, an RPQ R and two vertices s, t, is there a match to R in D that starts in s and ends in t?

Tuple Multiplicity – Given a graph database D, an RPQ R and two vertices s, t, how many matches to R in D start in s and end in t?

Walk Membership – Given a graph database D, an RPQ R and a walk w in D. Is w a match to R?

Walk Enumeration – Given a graph database D and an RPQ R, enumerate the **multiset**[2] of walks in D whose label conforms to R.

Distinct Walk Enumeration – Given a graph database D and an RPQ R, enumerate the **set** of walks in D whose label conforms to R.

6 Bibliography

[DFM23] Claire David, Nadime Francis, and Victor Marsault. "Run-Based Semantics for RPQs". In: *KR'23*. 2023. URL: https://arxiv.org/abs/2211.13313.

[Deu+22] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Hannes Voigt, Oskar van Rest, Domagoj Vrgoč, Mingxi Wu, and Fred Zemke. "Graph Pattern Matching in GQL and SQL/PGQ". In: *SIGMOD'22*. 2022. URL: https://arxiv.org/abs/2112.06217.

[2] In the data model we use, each edges in the graph can actually bear several labels. Hence the same walk might be a match to R multiple times.