Database Theory in Action: Cypher, GQL, and Regular Path Queries

Amélie Gheerbrant
Université Paris Cité, CNRS, IRIF, France

Leonid Libkin
RelationalAI and IRIF, CNRS, Paris, France
University of Edinburgh, UK

Liat Peterfreund
The Hebrew University of Jerusalem, Israel

Alexandra Rogova
Université Paris Cité, CNRS, IRIF, France

Abstract

Cypher has so far been the most commonly used query language for property graphs, and served as the foundation of the recently standardized graph query language GQL. In designing the features of GQL, the standards committee addressed the perceived limitations of Cypher. One such limitation is the inability of Cypher, as originally designed, to express all regular path queries (RPQs). Despite this claim having been stated many times as a folklore result, we could not find any proof of it. In this note we formalize the core of Cypher's pattern matching and formally prove that indeed it falls short of all RPQs, justifying the inclusion of new pattern matching features in GQL.

2012 ACM Subject Classification Information systems → Graph-based database models

Keywords and phrases Regular path queries, Cypher, GQL, inexpressibility

Digital Object Identifier 10.4230/LIPIcs.ICDT.2025.36

Funding We acknowledge support from: VeriGraph project, ANR-21-CE48-0015 (L. Libkin); Israel Science Foundation 2355/24 (L. Peterfreund); NCN grant 2018/30/E/ST6/00042 (A. Rogova).

1 Introduction

One of the key goals of the design of the new standard Graph Query Language (*ISO/IEC 39075:2024 Information technology – Database languages – GQL*, `www.iso.org/standard/76120.html`) was to overcome Cypher's perceived limitation with respect to regular path queries (RPQs). One finds statements that Cypher falls short of the full power of RPQs in many sources and surveys, e.g. [5, 2, 1]. There is a strong intuition behind this statement: in Cypher, the use of Kleene star $*$ is limited to edge labels. Essentially, one can say that there is a path of edges labeled ℓ between two nodes n and n', but it seems that one cannot say that there is a path $n = n_0 \cdots n_1 \cdots n_2 \cdots n_{k-1} \cdots n_k = n'$ so that each part of this path between n_i and n_{i+1} for $0 \leq i < k$, satisfies a pattern π more complex than a labeled edge.

This observation led to the substantial extension of GQL's pattern matching, which also coincide with available pattern matching in SQL/PGQ, a newly standardized extension of SQL with mechanisms for property graph querying (see their informal description in [3]). Namely, a bounded (between n and m times for $n < m \in \mathbb{N}$) or unbounded (at least n times) repetition can be applied to an *arbitrary* pattern π (currently such patterns cannot themselves contain repetitions, but a *language opportunity*[1] exists to remove this restriction).

[1] In SQL and GQL standards, this means the committee plans to revisit a specific feature.

This substantial extension of the language was based on a *belief* that arbitrary regular properties of paths cannot be expressed in Cypher. Our goal is to *justify* this decision by providing a *proof* of the widely believed – but hitherto unproven – statement.

2 Graph databases and Cypher patterns: an abstraction

Cypher operates on *property graphs*: these are labeled graphs, potentially with multiple edges between two nodes, where both edges and nodes carry properties given as key/value pairs. The latter play no role in comparison with RPQs which only refer to labels; thus we use a simple model of graph databases where properties are disregarded.

Graph databases. Assume pairwise disjoint countable sets $\mathcal{N}$ of node ids, $\mathcal{E}$ of edge ids and $\mathcal{L}$ of labels. A *graph database* is a tuple $G = \langle N, E, \lambda, \sigma, \tau \rangle$ where

- $N \subset \mathcal{N}$ is a finite set of node ids used in G;
- $E \subset \mathcal{E}$ is a finite set of directed edge ids used in G;
- $\lambda : N \cup E \to \mathcal{L}$ is a labeling function that associates with every id a label from $\mathcal{L}$;
- $\sigma, \tau : E \to N$ define source and target of an edge.

In real Cypher, λ can assign sets of labels to a node, and in GQL it can assign sets of labels to edges as well, but since the separating example does not depend on these features (which can also be modeled by considering $2^{\mathcal{L}}$ as the new set of labels), we do not use them here.

A *path* in G is an alternating sequence $n_0 e_1 n_1 e_2 \cdots e_k n_k$, for $k \geq 0$, of nodes and edges that starts and ends with a node and so that each edge e_i connects n_{i-1} to n_i for $i \leq k$. That is, either e_i is a forward edge with $\sigma(e_i) = n_{i-1}$ and $\tau(e_i) = n_i$, or a backward edgewith $\sigma(e_i) = n_i$ and $\tau(e_i) = n_{i-1}$. If $k = 0$, the path consists of a single node n_0. We explicitly write out paths as $\langle\!\langle n_0, e_1, n_1, \cdots, e_k, n_k \rangle\!\rangle$. Two paths $p = \langle\!\langle n_0, e_0, \ldots, n_k \rangle\!\rangle$ and $p' = \langle\!\langle n'_0, e'_0, \ldots, n'_j \rangle\!\rangle$ *concatenate*, if $n_k = n'_0$, in which case their *concatenation* $p \cdot p'$ is defined as $\langle\!\langle n_0, e_0, \ldots, n_k, e'_0, \ldots, n'_j \rangle\!\rangle$.

Cypher Patterns. We refine an abstraction of patterns of GQL from [4], omitting the GQL specific features, and adding Cypher patterns matching repeated edges of a given label. Let $\mathcal{V}$ be a countably infinite set of variables. Patterns are defined by

$$\begin{array}{rcllllll} \pi & := & (x:\ell) & | & (x) \xrightarrow{y:\ell} (z) & | & (x) \xleftarrow{y:\ell} (z) \;|\; (x) \xrightarrow{\ell^*} (z) \;|\; (x) \xleftarrow{\ell^*} (z) \\ & | & \pi_1\, \pi_2 & | & \pi_1 + \pi_2 & | & \pi\langle\theta\rangle, \qquad x, y, z \in \mathcal{V} \end{array}$$

Let $\mathcal{V}(\pi)$ be the set of all variables mentioned in π. The syntactic conditions are:

- *conditions* in $\pi\langle\theta\rangle$ are given by $\theta, \theta' := (x = x') \mid \theta \vee \theta' \mid \theta \wedge \theta' \mid \neg\theta$; all variables mentioned in θ must occur in $\mathcal{V}(\pi)$.
- $\pi_1 + \pi_2$ is only defined when $\mathcal{V}(\pi_1) = \mathcal{V}(\pi_2)$.

In Cypher variables and labels in edge/node patterns are optional, but we include them for simplicity. This does not affect the separating example in which we assume one fixed label for all nodes/edges; the use of variables does not affect expressibility of RPQs. Also, Cypher's repetitions of labeled edges of the form $n..m$ (between n and m) or $n..$ (at least n), or $..m$ (at most m) are all expressible with concatenation, union, and Kleene star.

Semantics of Patterns. The *semantics* $[\![\pi]\!]$ *of a path pattern* π, with respect to a graph database G, is a set of pairs (p, μ) where p is a path and μ is a mapping $\mathcal{V}(\pi) \to N \cup E$ as defined in Fig. 1. Two partial mappings $\mu, \mu' : \mathcal{V} \to N \cup E$ are *joinable* if $\mu(x) = \mu'(x)$

$$
\begin{array}{rcl}
[\![(x)]\!] & := & \{(\langle\!\langle n \rangle\!\rangle, \{x \mapsto n\}) \mid n \in N\} \\
\left[\!\!\left[\overset{x}{\rightarrow}\right]\!\!\right] & := & \{(\langle\!\langle n_1, e, n_2 \rangle\!\rangle, \{x \mapsto e\}) \mid e \in E,\ \sigma(e) = n_1,\ \tau(e) = n_2\} \\
\left[\!\!\left[\overset{x}{\leftarrow}\right]\!\!\right] & := & \{(\langle\!\langle n_2, e, n_1 \rangle\!\rangle, \{x \mapsto e\}) \mid e \in E,\ \sigma(e) = n_1,\ \tau(e) = n_2\} \\
\left[\!\!\left[(x) \xrightarrow{:\ell*} (z)\right]\!\!\right] & := & \{(\langle\!\langle n_1, e_1, n_2, \ldots, e_{k-1}, n_k \rangle\!\rangle, \{x \mapsto n_1, z \mapsto n_k\}) \mid \\
 & & e_1, \ldots, e_{k-1} \in E, \sigma(e_i) = n_i, \tau(e_i) = n_{i+1}, \lambda(e_i) = \ell \text{ for all } i < k\} \\
\left[\!\!\left[(x) \xleftarrow{:\ell*} (z)\right]\!\!\right] & := & \{(\langle\!\langle n_1, e_1, n_2, \ldots, e_{k-1}, n_k \rangle\!\rangle, \{x \mapsto n_k, z \mapsto n_1\}) \mid \\
 & & e_1, \ldots, e_{k-1} \in E, \sigma(e_i) = n_{i+1}, \tau(e_i) = n_i, \lambda(e_i) = \ell \text{ for all } i < k\} \\
[\![\pi_1 + \pi_2]\!] & := & [\![\pi_1]\!] \cup [\![\pi_2]\!] \\
[\![\pi_1\, \pi_2]\!] & := & \{(p_1 \cdot p_2, \mu_1 \bowtie \mu_2) \mid (p_1, \mu_1) \in [\![\pi_1]\!],\ (p_2, \mu_2) \in [\![\pi_2]\!], \\
 & & \mu_1, \mu_2 \text{ are joinable and } p_1, p_2 \text{ concatenate}\} \\
[\![\pi\langle\theta\rangle]\!] & := & \{(p, \mu) \in [\![\pi]\!] \mid \mu \models \theta\} \quad \text{where } \mu \models x = y \text{ iff } \mu(x) = \mu(y)
\end{array}
$$

Figure 1 Semantics of Cypher patterns with respect to $G = \langle N, E, \lambda, \sigma, \tau \rangle$.

for each shared x. Their join $\mu \bowtie \mu'$ is then unambiguously defined as the mapping that coincides with $\mu(x)$ for x in the domain of μ and $\mu'(x)$ for x in the domain of μ'. In the figure, we omit the standard conditions for $\mu \models \theta$ for Boolean connectives.

3 Cypher Vs. RPQs

Recall that an RPQ is a regular expression q over the labels $\mathcal{L}$. The result of q in G, written $q(G)$, is the set of pairs of nodes (n_0, n_k) such that there is a path $\langle\!\langle n_0, e_0, n_1, e_1, \ldots, e_{k-1}, n_k \rangle\!\rangle$ with the word $\lambda(e_0)\lambda(e_1)\cdots\lambda(e_{k-1})$ being in the regular language of q.

A Cypher pattern π with two designated variables $x_s, x_t \in \mathcal{V}(\pi)$ is said to express an RPQ q in G if $q(G) = \{(\mu(x_s), \mu(x_t)) \mid (p, \mu) \in [\![\pi]\!]\}$.

▶ **Theorem 1.** *Cypher patterns, as defined above, cannot express all RPQs. In particular they cannot express the pattern testing for an even-length path of edges labeled ℓ.*

In other words, the regular path query $(\ell\ell)^*$ cannot be expressed in Cypher.

Proof. Consider graphs G_n with $N = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_{n-1}\}$ so that $\sigma(e_i) = v_i$ and $\tau(e_i) = v_{i+1}$ for $i < n$ (i.e., directed paths), with each edge labelled ℓ. We can therefore assume that all edge labels used in patterns are ℓ (if not, such a pattern is not matched, and thus the entire subpattern in which it occurred cannot be matched, up to $+$ in the parse tree, and thus can be removed). We can also further assume that no variable is used as both a node variable and an edge variable (as this would falsify the pattern), nor any explicit equality between such variables is used in conditional patterns.

We now represent such graphs G_n as first-order structures S_n in the vocabulary R, R^* with the universe N, and relations interpreted as follows:

- $R = \{(v_i, v_{i+1}) \mid 1 \leq i < n\}$ is the edge relation;
- $R^* = \{(v_i, v_j) \mid 1 \leq i \leq j \leq n\}$ is the reflexive transitive closure of R.

We next show how patterns are translated into first-order formulae over this vocabulary. We use R for convenience, as it is definable from R^* which is isomorphic to a linear order on $\{1, \ldots, n\}$. We will then easily obtain the inexpressibility results since FO cannot define even cardinality of linear orders.

For the translation, with each pattern π we associate two new variables x^s_π, x^t_π (intuitively, to be witnessed by the endpoints of patterns), and with each edge variable z used in a pattern we associate two first-order variables z^s, z^t to be used in FO formulas (for source and target of edges). Then a pattern π with node variables $y_1, \ldots, y_m$ and edge variables $z_1, \ldots, z_k$ (all distinct) is translated into an FO formula $\alpha_\pi(x^s_\pi, x^t_\pi, y_1, \ldots, y_m, z^s_1, z^t_1, \ldots, z^s_k, z^t_k)$. The condition on the translation is that for a path $p = \langle\langle u_0, f_0, u_1, \ldots, f_{r-1}, u_r\rangle\rangle$ we have

$$\begin{array}{ll} & (p,\mu) \in [\![\pi]\!]_{G_n} \\ \Leftrightarrow & S_n \models \alpha_\pi\big(u_0, u_r, \mu(y_1), \ldots, \mu(y_m), \sigma(\mu(z_1)), \tau(\mu(z_1)), \ldots, \sigma(\mu(z_k)), \tau(\mu(z_k))\big) \end{array} \tag{1}$$

Now suppose the pattern $(\ell\ell)^*$ is definable in Cypher over graphs G_n by a pattern π as above. Then $\beta(x^s_\pi, x^t_\pi) := \exists y_1, \ldots, y_m, z^s_1, \ldots, z^t_k\ \alpha_\pi$ is true for v_i, v_j iff the path between them is of even length and therefore the sentence $\gamma := \exists s, t\ \big(\neg\exists s'\ R(s', s) \wedge \neg\exists t'\ R(t, t') \wedge \beta(s, t)\big)$ states the path from v_1 to v_n is of even length, which is impossible.

Next, to conclude the proof, we present the translation, recursively.

- If $\pi = (y)$ then $\alpha_\pi(x^s_\pi, x^t_\pi, y) := x^s_\pi = x^t_\pi \wedge x^t_\pi = y$.
- If $\pi = (y_1) \overset{z:\ell}{\rightarrow} (y_2)$ then $\alpha_\pi(x^s_\pi, x^t_\pi, y_1, y_2, z^s, z^t) := x^s_\pi = y_1 \wedge x^t_\pi = y_2 \wedge z^s = y_1 \wedge z^t = y_2 \wedge R(y_1, y_2)$.
- If $\pi = (y_1) \overset{z:\ell}{\leftarrow} (y_2)$ then $\alpha_\pi(x^s_\pi, x^t_\pi, y_1, y_2, z^s, z^t) := x^s_\pi = y_2 \wedge x^t_\pi = y_1 \wedge z^s = y_2 \wedge z^t = y_1 \wedge R(y_2, y_1)$.
- If $\pi = (y_1) \overset{:\ell^*}{\rightarrow} (y_2)$ then $\alpha_\pi(x^s_\pi, x^t_\pi, y_1, y_2) := x^s_\pi = y_1 \wedge x^t_\pi = y_2 \wedge R^*(y_1, y_2)$
- if $\pi = (y_1) \overset{:\ell^*}{\leftarrow} (y_2)$ then $\alpha_\pi(x^s_\pi, x^t_\pi, y_1, y_2) := x^s_\pi = y_2 \wedge x^t_\pi = y_1 \wedge R^*(y_2, y_1)$.
- If $\pi = \pi_1\pi_2$ with π_1, π_2 translated as $\alpha_{\pi_1}(x^s_{\pi_1}, x^t_{\pi_1}, \overline{v}_1)$ and $\alpha_{\pi_2}(x^s_{\pi_2}, x^t_{\pi_2}, \overline{v}_2)$ respectively (where $\overline{v}_i$ list variables in those formulae corresponding to node and edge variables in patterns), then $\alpha_\pi(x^s_\pi, x^t_\pi, \overline{v}_1, \overline{v}_2)$ is defined as

  $$\exists x^s_{\pi_1}, x^t_{\pi_1}, x^s_{\pi_2}, x^t_{\pi_2}\ \Big(\alpha_{\pi_1}(x^s_{\pi_1}, x^t_{\pi_1}, \overline{v}_1) \wedge \alpha_{\pi_2}(x^s_{\pi_2}, x^t_{\pi_2}, \overline{v}_2) \wedge x^s_\pi = x^s_{\pi_1} \wedge x^t_\pi = x^t_{\pi_2} \wedge x^t_{\pi_1} = x^s_{\pi_2}\Big)$$

  where in $\overline{v}_1, \overline{v}_2$ repeated variables are mentioned only once.
- If $\pi = \pi_1 + \pi_2$ with π_1, π_2 translated as $\alpha_{\pi_1}(x^s_{\pi_1}, x^t_{\pi_1}, \overline{v})$ and $\alpha_{\pi_2}(x^s_{\pi_2}, x^t_{\pi_2}, \overline{v})$ (note that variables must be the same as the schemas of π_1 and π_2 coincide), then $\alpha_\pi(x^s_\pi, x^t_\pi, \overline{v}) := \alpha_{\pi_1}(x^s_\pi, x^t_\pi, \overline{v}) \vee \alpha_{\pi_2}(x^s_\pi, x^t_\pi, \overline{v})$.
- If $\pi = \pi_1\langle\theta\rangle$ then $\alpha_\pi(x^s_\pi, x^t_\pi, \overline{v}) := \alpha_{\pi_1}(x^s_\pi, x^t_\pi, \overline{v}) \wedge \theta'$ where θ' is obtained from θ by the following transformations:
  - each condition $y_i = y_j$ stays;
  - each condition $z_i = z_j$ is replaced by $z^s_i = z^s_j \wedge z^t_i = z^t_j$;
  - these are propagated through the Boolean connectives.

It is straightforward to verify that these translations satisfy (1), completing the proof. ◀

The proof shows that on graphs G_n, Cypher patterns fall *far* short of RPQs. The latter can express every regular property of languages in ℓ^*, or in other words test if n belongs to a set which is a finite union of arithmetic progressions. For Cypher patterns, on the other hand, the first-order definability of a pattern π in the theory of order implies the existence of the threshold t such that either (v_1, v_n) is selected by π for all $n > t$, or (v_1, v_n) is not selected by π for all $n > t$.

4 Conclusions

With formal models of GQL, SQL/PGQ, and Cypher finally available, this note is an example of how research in database theory can affect the design of new languages. When it comes to graph languages, industrial developments are far ahead of academic research, creating opportunities for the academic community to develop tools to evaluate decisions already made and lay a solid foundation for new language features in upcoming editions of the standards.

References

1 Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5):68:1–68:40, 2017. doi:10.1145/3104031.

2 Angela Bonifati, George H. L. Fletcher, Hannes Voigt, and Nikolay Yakovets. *Querying Graphs*. Morgan & Claypool Publishers, 2018. doi:10.2200/S00873ED1V01Y201808DTM051.

3 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*, pages 1–12. ACM, 2022. arXiv:2112.06217.

4 Nadime Francis, Amélie Gheerbrant, Paolo Guagliardo, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Liat Peterfreund, Alexandra Rogova, and Domagoj Vrgoc. GPC: A pattern calculus for property graphs. In Floris Geerts, Hung Q. Ngo, and Stavros Sintos, editors, *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 241–250. ACM, 2023. doi:10.1145/3584372.3588662.

5 Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1433–1445, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3183713.3190657.