DATABASE THEORY

Lecture 17: Regular Path Queries

David Carral
Knowledge-Based Systems

TU Dresden, June 15, 2020

Review: Graph databases

Graph databases emphasise connections within databases
$\rightsquigarrow$ connectivity, paths, and network structure play an important role

Resource Description Framework:

- Directed, labelled graphs that use URIs and standard datatypes
- Implemented in many programming languages and systems
- W3C standard for graph-based data exchange on the Web
- SPARQL as query language

Property graph:

- Directed multi-graphs, labelled with attribute-value sets
- Based on common implementation (TinkerPop, Java), used in many systems
- Defined by reference implementation; no exchange syntax
- Cypher and others as query languages

Either model can fully capture the other for representing graphs
(conceptually speaking; the exact behaviour of specific datatypes and queries may not always be easy to translate)

Conjunctive Queries over Graphs

Basic descriptions of local patterns in a graph

Formally, it suffices to say:

"CQs over RDF correspond to CQs over relational databases with a single table Triple[Subject,Predicate,Object]"

(and analogously for Property Graphs)

- All complexity results for query answering and optimisation carry over from RDBs (in particular, restricting to graphs does not make anything simpler)
- Details of representation of data in tables do not matter
- CQs are restricted to local patterns (no reachability . . .)

Regular Path Queries

Idea: use regular expressions to navigate over paths

Let's consider a simplified graph model, where a graph is given by:

- Set of nodes N (without additional labels)
- Set of edges E, labelled by a function $\lambda : E \to L$, where L is a finite set of labels

Definition 17.1: A regular expression over a set of labels L is an expression of the following form:

$$E ::= L \mid (E \circ E) \mid (E + E) \mid E^*$$

A regular path query (RPQ) is an expression of the form $E(s, t)$, where E is a regular expression and s and t are terms (constants or variables).

Semantics of Regular Path Queries

As usual, a regular expression E matches a word $w = \ell_1 \cdots \ell_n$ if any of the following conditions is satisfied:

- $E \in L$ is a label and $w = E$.
- $E = (E_1 \circ E_2)$ and there is $i \in \{0, \ldots, n\}$ such that E_1 matches $\ell_1 \cdots \ell_i$ and E_2 matches $\ell_{i+1} \cdots \ell_n$ (the words matched by E_1 and E_2 can be empty if $i = 0$ or $i = n$, respectively).
- $E = (E_1 + E_2)$ and w is matched by E_1 or by E_2
- $E = E_1^*$ and w has the form $w_1 w_2 \cdots w_m$ for $n \geq 0$, where each word w_i is matched by E_1

Definition 17.2: Let a and b be constants and x and y be variables. An RPQ $E(a, b)$ is entailed by a graph G if there is a directed path from node a to node b that is labelled by a word matched by E. The answers to RPQs $E(x, y)$, $E(x, b)$, and $E(a, y)$ are defined in the obvious way.

Extending the Expressive Power of RPQs

Regular path queries can be used to express typical reachability queries, but are still quite limited $\leadsto$ extensions

2-Way Regular Path Queries (2RPQs)

- For every label $\ell \in L$, also introduce a converse label ℓ^-
- Allow converse labels in regular expressions
- Matched paths can follow edges forwards or backwards

Conjunctive Regular Path Queries (CRPQs)

- Extend conjunctive queries with RPQs
- RPQs can be used like binary query atoms
- Obvious semantics

Conjunctive 2-Way Regular Path Queries (C2RPQs) combine both extensions

C2RPQs: Examples

All ancestors of Alice:

$$((\text{father} + \text{mother}) \circ (\text{father} + \text{mother})^*)(\text{alice}, y)$$

People with finite Erdös number:

$$(\text{authorOf} \circ \text{authorOf}^-)^*(x, \text{paulErdös})$$

Pairs of stops connected by tram lines 3 and 8:

$$(\text{nextStop3} \circ \text{nextStop3}^*)(x, y) \land (\text{nextStop8} \circ \text{nextStop8}^*)(x, y)$$

Complexity of RPQs

A nondeterministic algorithm for Boolean RPQs:

- Transform regular expression into a finite automaton
- Starting from the first node, guess a matching path
- When moving along path, advance state of automaton
- Accept if the second node is reached in an accepting state
- Reject if path is longer than size of graph × size of automaton

Space requirements when assuming query (and automaton) fixed: pointer to current node in graph, pointer to current state of automaton, counter for length of path $\leadsto$ NL algorithm

Conversely, reachability in an unlabelled graph is hard for NL $\leadsto$ RPQ matching is NL-complete (data complexity)

(Combined/query complexity is in P, as we will see below)

Complexity of C2RPQs

We already know:

- CQ matching is in AC^0 (data complexity) and NP-complete (query and combined complexity)
- RPQ matching is NL-complete (data) and in P (query/combined)
- $AC^0 \subset$ NL and NL $\subseteq$ NP

$\leadsto$ C2RPQs are NP-hard (combined/query) and NL-hard (data)

It's not hard to show that these bounds are tight:

Theorem 17.3: C2RPQ matching is NP-complete for combined and query complexity, and NL-complete for data complexity.

(C2)RPQs and Datalog

How do path queries relate to Datalog?

We already know:

- Datalog is ExpTime-complete (combined/query) and P-complete (data)
- C2RPQs are NP-complete (combined/query) and NL-complete (data)

$\leadsto$ maybe Datalog is more expressive that C2RPQs ...

Indeed, we can express regular expressions in Datalog

For simplicity, assume that we have a binary EDB predicate p_ℓ for each label $\ell \in L$ (other encodings would work just as well)

2-Way Regular Expressions in Datalog

We transform a regular expression E to a Datalog query $\langle Q_E, P_E\rangle$:

If $E = \ell \in L$ is a label, then $P_E = \{\mathsf{Q}_E(x,y) \leftarrow \mathsf{p}_\ell(x,y)\}$

If $E = \ell^-$ is the converse of a label $\ell \in L$, then

$$P_E = \{\mathsf{Q}_E(x,y) \leftarrow \mathsf{p}_\ell(y,x)\}$$

If $E = (E_1 \circ E_2)$ then

$$P_E = P_{E_1} \cup P_{E_2} \cup \{\mathsf{Q}_E(x,z) \leftarrow \mathsf{Q}_{E_1}(x,y) \wedge \mathsf{Q}_{E_2}(y,z)\}$$

If $E = (E_1 + E_2)$ then

$$P_E = P_{E_1} \cup P_{E_2} \cup \{\mathsf{Q}_E(x,y) \leftarrow \mathsf{Q}_{E_1}(x,y), \mathsf{Q}_E(x,y) \leftarrow \mathsf{Q}_{E_2}(x,y)\}$$

If $E = E_1^*$ then

$$P_E = P_{E_1} \cup \{\mathsf{Q}_E(x,x) \leftarrow, \mathsf{Q}_E(x,z) \leftarrow \mathsf{Q}_E(x,y) \wedge \mathsf{Q}_{E_1}(y,z)\}$$

Reprise: Combined Complexity of 2RPQs

As a side effect, the previous translation shows that 2RPQs can be evaluated in P combined complexity:

- Each (2-way) regular expression E leads to a Datalog query $\langle Q_E, P_E\rangle$ of polynomial size
- Each rule in P_E has at most three variables
  $\leadsto$ the grounding of P_E for a graph with nodes N is of size $|P_E| \times |N|^3$
- propositional logic rules can be evaluated in polynomial time

$\leadsto$ polynomial time decision procedure

Expressing C2RPQs in Datalog

It is now easy to express C2RPQs in Datalog:

- Use the encoding of CQs in Datalog as shown in the exercise
- Express 2RPQ atoms in Datalog as just shown

Can every Datalog query over binary "labelled-edge" EDB predicates be expressed with (C2)RPQs?

- This would imply P = NL (but not that NP = ExpTime!): unlikely but not known to be false
- However, there are stronger direct arguments that show the limits of C2RPQs (exercise)

Linear Datalog and Binary Datalog

Expressing 2RPQs in Datalog requires only restricted forms of Datalog:

Definition 17.4: A Datalog program is linear if each of its rules has at most one IDB atom in its body. A Datalog program is binary if all of its IDB predicates have arity at most two.

The following complexity results are known:

Theorem 17.5: Query answering in linear Datalog is NL-complete for data complexity, and PSpace-complete for combined and query complexity.
Combined complexity further drops to NP for binary Datalog.

$\leadsto$ complexity results that are more similar to (C2)RPQs . . .

2RPQs and Linear Datalog

The Datalog translation of 2RPQs does not lead to linear Datalog, but we can fix this.

We transform a regular expression E to a linear Datalog query $\langle \mathsf{Q}_E, P_E^{\mathsf{lin}} \rangle$:

- Construct a non-deterministic automaton $\mathcal{A}_E$ for E
- For every state q of $\mathcal{A}_E$, we use a binary IDB predicate S_q
- For the starting state q_0 of $\mathcal{A}_E$, we add a rule $\mathsf{S}_{q_0}(x, x) \leftarrow$
- For every transition $q \xrightarrow{\ell} q'$ of $\mathcal{A}_E$, we add a rule

$$\mathsf{S}_{q'}(x, z) \leftarrow \mathsf{S}_q(x, y) \wedge \mathsf{p}_\ell(y, z)$$

- For every final state q_f of $\mathcal{A}_E$, we add a rule

$$\mathsf{Q}_E(x, y) \leftarrow \mathsf{S}_{q_f}(x, y)$$

Two-way queries can be captured by allowing two-way transitions.

Linear Datalog vs. 2RPQs

So all 2RPQs can be expessed in linear Datalog
Is the converse also true?

No. Counterexample:

$$\mathsf{Query}(x, z) \leftarrow \mathsf{p}_a(x, y) \wedge \mathsf{p}_b(y, z)$$
$$\mathsf{Query}(x, z) \leftarrow \mathsf{p}_a(x, x') \wedge \mathsf{Query}(x', z') \wedge \mathsf{p}_b(z', z)$$

The linear Datalog program matches paths with labels from $a^n b^n$
$\leadsto$ context-free, non-regular language
$\leadsto$ not expressible in (C2)RPQs

Intuition: linear Datalog generalises context-free languages

Query Optimisation for C2RPQs

Recall the basic static optimisation problems of database theory:

- Query containment
- Query equivalence
- Query emptiness

Which of these are decidable for (C2)RPQs?

Observation: query emptiness is trivial

Containment for RPQs

Containment of Regular Path Queries corresponds to containment of regular expressions ⇝ known to be decidable in PSpace

Proof sketch for checking $E_1 \sqsubseteq E_2$:

(1) Construct non-deterministic automata (NFAs), A_1 and A_2 for the regular expressions E_1 and E_2, respectively

(2) Construct an automaton $\bar{A}_2$ that accepts the complement of A_2.

(3) Construct the intersection $A_1 \cap \bar{A}_2$ of A_1 and $\bar{A}_2$

(4) Check if $A_1 \cap \bar{A}_2$ accepts a word (if yes, then there is a counterexample that disproves $E_1 \sqsubseteq E_2$; if no, then the containment holds)

Complexity estimate:
$A_1 \cap \bar{A}_2$ is exponential (blow-up by powerset construction in step (2)) but step (4) is possible by checking reachability on the state graph
⇝ NL algorithm on an exponential state graph
⇝ NPSpace algorithm (construct the state graph on the fly)
⇝ PSpace algorithm (Savitch's Theorem)

Containment for (C)2RPQs

Things are more tricky when adding converses and conjunctions

Theorem 17.6:

- Containment of 2RPQs is PSpace-complete
- Containment of C2RPQs is ExpSpace-complete

The proofs are more involved.

Automata-theoretic constructions are used, but with more complicated automata models and for somewhat different languages (there is no good "language of possible C2RPQ matches on a graph" ⇝ consider language of possible proofs instead)

Query Optimisation for Path Queries

Decidable in PSpace (2RPQs) and ExpSpace (C2RPQs)

Should be compared to linear Datalog:

Theorem 17.7: Query containment for linear Datalog queries is undecidable.

Proof: see Lecture 13 (Post Correspondence Problem in Datalog – in fact, in linear Datalog) □

Essentially no adoption in practice
⇝ maybe the complexities are too high . . .
⇝ or maybe path query optimisers are just too primitive

Path Queries: Final Remarks on Expressivity

We have seen that C2RPQs are NL-complete for data
$\leadsto$ can all NL-complete queries be captured by a C2RPQ?

No. For many reasons.

- C2RPQs have no disjunction ($\leadsto$ Unions of C2RPQs)
- C2RPQs have no negation

FO-queries with a binary transitive closure operator capture NL

Several (regular) extensions of path queries:

- Nested unary 2RPQs in regular expressions ("test operators")
- Nested binary C2RPQs in regular expressions
- Other more expressive fragments of "regular Datalog", e.g., Monadically Defined Queries

Summary and Outlook

Regular Path Queries (RPQs) and their generalisation 2RPQs and C2RPQs define practically useful types of recursive queries

(2)RPQ answering is NL-complete (data) and P-complete (combined/query); query complexity goes up to NP for C2RPQs

Path queries can be expressed in linear Datalog, which is more expressive though

Query containment is decidable for path queries, but not for linear Datalog

Next topics:

- Logical dependencies
- Query answering under constraints