Regular path queries on graphs with data: A rigid approach

Zhilin Wu

State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
Background

- **Graph databases**
  - $G = (V, E)$
    - $V$: set of nodes,
    - $E \subseteq V \times \Sigma \times V$: set of labeled edges
Background

- Regular path queries (RPQ) and Conjunctive RPQ (CRPQ) [Buneman et al. 1996, 1997]
  - Specify regular path constraints between nodes
    \[ Q = (x, r, y), \text{ where } r \text{ is a regular expression} \]
  - CRPQ: \[ \text{res}(z_1, \ldots, z_k) : - (x_1, r_1, y_1) \land \ldots \land (x_k, r_k, y_k) \]
  - Examples
    - “\( x \) and \( y \) have the same father”
      \[ Q = (x, \text{ft}^{-} \text{ft}, y), \]
    - “\( x \) and \( y \) have the same parents”
      \[ \text{res}(x, y) : - (x, \text{ft}^{-} \text{ft}, y) \land (x, \text{mt}^{-} \text{mt}, y) \]
Background

- Regular data path queries (RDPQ) and Conjunctive RDPQ (CRDPQ) [Libkin & Vrgoc 2012]
  - Data graph $G=(V, E, \eta)$, where $\eta: V \rightarrow D$
Background

• Regular data path queries (RDPQ) and Conjunctive RDPQ (CRDPQ) [Libkin & Vrgoc 2012]

  – Q=(x, r, y), where r is
    a regular expression with memory (REM)
    REM ≡ Nondeterministic register automata

  – Examples

    “x is the maternal ancestor of y and
    for every edge (z, mt, z’) between them,
    z and z’ have different family names”

    Q=(x, fn ↓_1 fn^- (mt fn [¬↑_1] ↓_1 fn^-)^+, y)
Motivation

• Two drawbacks of RDPQs (REMs)
  – Imperativeness in dealing with data values
  – Undecidability of containment problem

• Regular exp. with rigid data constraints (RER)
  and Rigid regular data path queries (RRDPQ)
  – Declarativeness
  – Decidability of containment problem
Motivation

“x is the paternal ancestor of y and for every edge (z, mt, z’) between them, z and z’ have the same family name”

\[ Q = (x, fn \downarrow_1 fn^- (mt fn [\neg \uparrow_1] \downarrow_1 fn^-)^+, y) \]

\[ Q’ = (x, fn \downarrow_1 fn^- (mt fn [\neg (cur \sim pred_{fn^-})] fn^-)^+, y) \]

Rigid data constraints
Rigid data constraints

• Position terms

\[ t := \text{cur} \mid \text{pred}_A(t) \mid \text{suc}_A(t), \]

where \( A \subseteq \Sigma^\pm \)

Interpreted over data paths \( d_0 a_1 d_1 \ldots a_n d_n \)

– \text{cur}: the current position,

– \text{pred}_A(t) \text{ and } \text{suc}_A(t):

  the position immediately before (resp. after)

  the \textit{previous} (resp. \textit{next}) occurrence

  of letters from \( A \) before (resp. after) the position \( t \)
Rigid data constraints

- Rigid data constraints
  \[ c := t_1 \sim t_2 \mid \neg (t_1 \sim t_2) \mid c \lor c \mid c \land c \]

Example

\[ \alpha = 1 \ a \ 2 \ b \ 1 \ a \ 4 \ a \ 2 \ b \ 3 \]
\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \]

Let \( c = \neg (\text{cur} \sim \text{pred}_a(\text{suc}_b(\text{cur}))) \)

\[ (\text{pred}_a(\text{suc}_b(\text{cur})))\alpha[4]=\text{pred}_a[10]=6 \Rightarrow \]

\( (\alpha, 4) \mid = c \) since \( d_4=1 \neq 4=d_6. \)
Rigid regular data path queries (RRDPQ)

- Regular expressions with rigid data constraints (RER)
  \[ r := \emptyset | \epsilon | a | [c] | r \cdot r | r \cup r | r^* , \]
  where \( c \): rigid data constraint

- RRDPQ \( Q=(x, r, y) \), where \( r \) is a RER

Example

\[ Q=(x, \text{fn}\text{fn}^- (\text{mt} \text{fn}[\neg (\text{cur} \sim \text{pred}_{\text{fn}^-})] \text{fn}^-)^+, y) \]
Expressibility of RRDPQ

- In graph databases where all attributes are promoted as nodes (e.g. “given name” and “family name”)

RDPQ Q: One register for each attribute

⇒ RRDPQ Q’

Example:

Q = (x; fn ↓₁ fn⁻ (mt fn [¬ ↑₁] ↓₁ fn⁻)⁺, y) ⇒

Q’ = (x; fn fn⁻ (mt fn [¬ (cur ~ pred fn⁻)] fn⁻)⁺, y)
Expressibility of RRDPQ

• In general,
  RDPQ Q on G
  ⇒ RRDPQ Q’ on G’

• G ⇒ G’: By localized transformation
  Suppose k registers are used in Q, then
  for every v in G, add \( n_v \) and \( k \) edges,
  \((v, R_1, n_v), \ldots, (v, R_k, n_v)\)
Decision problems

- **Evaluation problem**
  - RRDPQ: the same as RDPQ
    - Combined complexity: PSPACE-c, Data complexity: NLOGSPACE-c
  - CRRDPQ: the same as CRDPQ
    - Combined complexity: PSPACE-c, Data complexity: NLOGSPACE-c

- **Containment problem**
  - RRDPQ: In EXPSPACE, PSPACE-hard
  - CRRDPQ: In 2EXPSPACE, EXPSPACE-hard
Finite state automata with rigid data constraints (NFAR)

- RER $\equiv$ NFAR
- Theoretical properties of NFAR
  - Closed under complementation
  - NFAR can be determinized
  - Two-way NFAR $=$ One-way NFAR
  - Decision problems of RER (NFAR)
    - Nonemptiness: PSPACE-c
    - language inclusion: PSPACE-c
Summary & Future work

• (Conjunctive) Rigid regular data path queries (RRDPQ and CRRDPQ)
  – Rigid data constraints: \( \text{pred}_A(t), \text{suc}_A(t) \)

• Compared to RDPQ
  – Declarativity
  – A good balance between expressiveness and decidability/complexity
    • Expressiveness:
      \( \text{RDPQ on } G \Rightarrow \text{RRDPQ on } G' \)
    • Containment of RDPQ and CRDPQ: Decidable
Summary & Future work

• Future work
  – Complexity gap: Containment of (C)RRDPQs
  – Extensions of RRDPQs
    Similar to nested regular expressions or XPath?
  – Push forward further the rigidity idea:
    More powerful position terms
    e.g. $\text{pred}_\varphi(t)$, $\text{suc}_\varphi(t)$, where
    $\varphi(x)$ is a MSO formula on words?