Regular Path Queries and Constraints
CSC2428 – Foundations of XML

Pablo Barceló

1 Regular path queries

A *semistructured* database (I, o) is composed by:

- a directed graph that is labeled over a finite alphabet Σ, and

- a *source* element o.

Notice that this is not necessarily a tree.

Given a regular expression $\mathcal{R}$ over Σ, the *path query* $\mathcal{R}$ is the function:

$$\mathcal{R} : (I, o) \rightarrow 2^I ,$$

such that $\mathcal{R}(I, o)$ is

$$\{ o' \in I \mid \text{there is a path labeled } a_1, \ldots, a_m \text{ from } o \text{ to } o', \text{such that } (a_1, \ldots, a_m) \in L(\mathcal{R})\} .$$

Nice way to evaluate path queries in *linear monadic* Datalog. Let $(Q, s, \Sigma, F, \delta)$ be any finite state automaton that computes $L(\mathcal{R})$. The extensional predicates are source(x) and $E(x, a, y)$ for each $a \in \Sigma$. Then path query $\mathcal{R}$ is computable by:

$$\begin{array}{rcll} \text{state}_s(x) & :- & \text{source}(x) & \\ \text{state}_h(x) & :- & \text{state}_j(y),\ E(y, a, x) & \forall a \in \Sigma, \text{ and } \forall j \in Q \text{ with } \delta(j, a) = h \\ \text{ans}(x) & :- & \text{state}_f(x) & \forall f \in F \end{array}$$

First, the complexity of evaluating a linear Datalog program is in NC, that is, the small parallel complexity class that contains families of circuits with polynomial number of gates, polylogarithmic depth, and constant fan-in. Second, monadic datalog programs allow nice optimization techniques to be used.

2 Regular path constraints

A *regular path inclusion* is of the form $\mathcal{R} \subseteq \mathcal{R}'$, for $\mathcal{R}, \mathcal{R}'$ regular expressions over Σ. Then

$$(I, o) \models \mathcal{R} \subseteq \mathcal{R}' \iff \mathcal{R}(I, o) \subseteq \mathcal{R}'(I, o) .$$

If $\mathcal{R}$ and $\mathcal{R}'$ are *words*, that is, sequences of symbols in Σ, then $\mathcal{R} \subseteq \mathcal{R}'$ is a *word constraint*. If E is a set of regular path inclusions, then $(I, o) \models E$ iff $(I, o) \models \mathcal{R} \subseteq \mathcal{R}'$ for each $\mathcal{R} \subseteq \mathcal{R}'$ in E.

We write $E \models \mathcal{R} \subseteq \mathcal{R}'$ iff for each (I, o),

$$(I, o) \models E \implies (I, o) \models \mathcal{R} \subseteq \mathcal{R}' .$$

Theorem 1 (Abiteboul and Vianu,'97) *It is decidable in 2-EXPSPACE (in the number of inclusions in E) whether $E \models \mathcal{R} \subseteq \mathcal{R}'$.*

Not very nice complexity, and not known if it can be improved. But,

Theorem 2 (Abiteboul and Vianu,'97) *If both E and $\mathcal{R} \subseteq \mathcal{R}'$ contain only word constraints, then it is polynomial to check whether $E \models \mathcal{R} \subseteq \mathcal{R}'$. Also, if E is a set of path constraints and $\mathcal{R} \subseteq \mathcal{R}'$ is a word constraint, then checking whether $E \models \mathcal{R} \subseteq \mathcal{R}'$ is in PSPACE.*

3 Extended path constraints

A *path* is a FO formula $\alpha(x, y)$ of one the following forms:

- $x = y$,
- $E(x, a, y)$ for $a \in \Sigma$, and
- $\exists z(E(x, a, x) \wedge \beta(z, y))$, where $a \in \Sigma$ and $\beta(z, y)$ is a path.

A *path constraint* is any expression of the form:

$$\begin{array}{ll} \forall x\,(\alpha(o,x) \;\rightarrow\; \forall y\,(\beta(x,y) \rightarrow \gamma(y,x))) & (\textit{backward}\text{ constraint}) \\ \forall x\,(\alpha(o,x) \;\rightarrow\; \forall y\,(\beta(x,y) \rightarrow \gamma(x,y))) & (\textit{forward}\text{ constraint}) \end{array}$$

where o denotes the source of I.

An example of a backward constraint is

$$\forall x\,(\text{Student}(o,x) \;\rightarrow\; \forall y\,(\text{Taking}(x,y) \rightarrow \text{Enrolled}(y,x)))\,.$$

Not expressible as a path inclusion constraint!

Theorem 3 (Buneman,Fan,Winstein,'98) *For E a set of path constraints, and ϕ a path constraint, it is undecidable to check if $E \models \phi$, even when we restrict to the finite case, and even if we restrict to the forward form.*

Nevertheless, if we denote by P_β the set of all path constraints such that either $\alpha \equiv \texttt{true}$, or β is of the form $x = y$ or $E(x, a, y)$, then

Theorem 4 (Buneman,Fan,Winstein,'98) *For E a set of path constraints in P_β, and ϕ a path constraint in P_β, it is decidable in EXPSPACE to check if $E \models \phi$.*

4 Regular path constraints with data values

Idea: *Extended keys* and *extended foreign keys* with regular expressions. That is, constraints of the form

$$\mathcal{R}.a.l \;\rightarrow\; \mathcal{R}.a \;\; \text{(keys)} \qquad \text{and} \qquad \mathcal{R}.a.l \;\subseteq \mathcal{R}'.a'.l' \;\; \text{(foreign keys)}$$

for $\mathcal{R}, \mathcal{R}'$ regular expressions in Σ, and $a, a' \in \Sigma$.

Keys are evaluated on trees as follows:

$$T \models \mathcal{R}.a.l \;\rightarrow\; \mathcal{R}.a \quad \Longleftrightarrow \quad \forall s, s' \in \mathcal{R}.a(T, \varepsilon),\; \text{if } (s.l = s'.l) \text{ then } (s = s')\,.$$

Foreign keys are combinations of inclusion dependencies and foreign keys, that is, $T \models \mathcal{R}.a.l \;\subseteq \mathcal{R}'.a'.l'$ iff $\mathcal{R}'.a'.l'$ is a key and

$$\forall s \in \mathcal{R}.a(T, \varepsilon),\; s.l = s'.l' \;\text{ for some } s' \in \mathcal{R}'.a'(T, \varepsilon)\,.$$

Theorem 5 (Arenas, Fan, Libkin,'02) *Checking for a set Σ of keys and foreign keys whether there is a tree T such that $T \models \Sigma$ is in NEXPTIME, and cannot be less than PSPACE.*

This shows that the complexity increases when having extended constraints. From (Fan,Libkin,'01), for *usual* keys and foreign keys the result is NP-complete.

5 Queries with data values

We consider the following fragment of XPath. Syntax for *path* queries p is given by:

$$p \;:=\; \epsilon \;\mid\; a,\; a \in \Sigma \;\mid\; \text{child} \;\mid\; \text{desc} \;\mid\; \text{parent} \;\mid\; \text{ancestor} \;\mid\; p/p \;\mid\; p \cup p \;\mid\; p[q]$$

where q is a *data value* expression given as follows:

$$q \;:=\; p \;\mid\; p/@a = c \;\mid\; p/@a = p/@b \;\mid\; q \wedge q \;\mid \neg q$$

A node s in a tree T satsifies a path query p iff there is s' such that $T \models p(s, s')$, where:

- if $p = \epsilon$ then $s = s'$,
- if $p = a$ then $s' = s$, and s is labeled a,
- axis are trivial,
- if $p = p_1/p_2$ then there is s'' such that $T \models p_1(s, s'')$ and $T \models p_2(s'', s')$,
- if $p = p_1 \cup p_2$ then either $T \models p_1(s, s')$ or $T \models p_2(s, s')$,
- if $p = p_1[q]$ then $T \models p_1(s, s')$ and
  - if $q = p_2$ then there is s'' such that $T \models p_2(s', s'')$,
  - if $q = (p_2/@a = c)$ then there is s'' such that $T \models p_2(s', s'')$ and $s''.a = c$,
  - if $q = (p_2/@a = p_3/@b)$ then there are s_1, s_2 such that $T \models p_2(s', s_1) \wedge p_3(s', s_2)$ and $s_1.a = s_2.b$,
  - Boolean combinations are trivial.

We write $\mathrm{SAT}(p, D)$ if there is a tree T that conforms to DTD D, and such that the output of p in T is nonempty.

Theorem 6 (Benedikt,Fan,Geerts,'05) *Checking for a DTD D and a path query p that uses neither concatenation / nor negations in data values expressions, whether* $\mathrm{SAT}(p, D)$*, is NP-complete. The same is true even for the fragment* $(\mathrm{child}, [\,])$ *without negation on data value expressions (even if DTDs are non-recursive).*

What if we admit negation?

Theorem 7 (Benedikt,Fan,Geerts,'05) *By admitting negation inside* $[\,]$ *we get undecidability. The fragment* $(\mathrm{child}, \cup, [\,])$ *is in NEXPTIME, while* $(\mathrm{parent}, [\,])$ *is already EXPTIME-hard.*

Several improvements can be found depending on simplifications on DTDs. What if we do not have DTDs?

Theorem 8 (Benedikt,Fan,Geerts,'05) *For the fragment* $(\cup, [\,])$ *without negations on data value expressions,* $\mathrm{SAT}(p, \emptyset)$ *is NP-hard. Furthermore,* $(\mathrm{parent}, [\,])$ *is EXPTIME-hard.*