Datalog Evaluation

Serge Abiteboul

INRIA

5 mai 2009
READ CHAPTER 13

lots of research in the late 80’th
top-down or bottom-up evaluation
direct evaluation vs. compilation into a more efficient program
no product
some influence on logic programming

HERE

1. semi-naive bottom-up evaluation
2. top-down : QSQ
3. bottom-up : Magic
Reverse-Same-Generation

\[
\begin{align*}
\text{rsg}(x, y) & \leftarrow \text{flat}(x, y) \\
\text{rsg}(x, y) & \leftarrow \text{up}(x, x1), \text{rsg}(y1, x1), \text{down}(y1, y)
\end{align*}
\]

\[\quad\]

<table>
<thead>
<tr>
<th>up</th>
<th>flat</th>
<th>down</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>g</td>
<td>l</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>a</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>f</td>
<td>n</td>
<td>f</td>
</tr>
<tr>
<td>m</td>
<td>o</td>
<td>b</td>
</tr>
<tr>
<td>g</td>
<td>m</td>
<td>g</td>
</tr>
<tr>
<td>n</td>
<td>p</td>
<td>h</td>
</tr>
<tr>
<td>h</td>
<td>m</td>
<td>c</td>
</tr>
<tr>
<td>i</td>
<td>p</td>
<td>i</td>
</tr>
<tr>
<td>o</td>
<td>k</td>
<td>d</td>
</tr>
<tr>
<td>j</td>
<td></td>
<td>p</td>
</tr>
</tbody>
</table>
Naive algorithm

level 0 : $\emptyset$
level 1 : \{(g, f), (m, n), (m, o), (p, m)\}
level 2 : \{level 1\} $\cup$ \{(a, b), (h, f), (i, f), (j, f), (f, k)\}
level 3 : \{level 2\} $\cup$ \{(a, c), (a, d)\}
level 4 : \{level 3\}

\[
\text{rsg} := \emptyset
\]
\[
\text{repeat}
\]
\[
\text{rsg} := \text{rsg} \cup \text{flat} \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(\text{up} \times \text{rsg} \times \text{down})))
\]
\[
\text{until fixpoint}
\]
\[
\text{rsg}^{i+1} = \text{rsg}^{i} \cup \text{flat} \cup \pi_{16}(\sigma_{2=4}(\sigma_{3=5}(\text{up} \times \text{rsg}^{i} \times \text{down})))
\]

redundant computation
each layer recomputes all elements of the previous layer
monotonicity
Semi-naive

Focus on the new facts generated at each level

RSG'

\[
\Delta^1_{rsg}(x, y) \leftarrow \text{flat}(x, y)
\]
\[
\Delta^{i+1}_{rsg}(x, y) \leftarrow \text{up}(x, x1), \Delta^i_{rsg}(y1, x1), \text{down}(y1, y)
\]

not recursive

not a datalog program

for each input I,

\[
rsg^{i+1} - rsg^i \subseteq \delta^{i+1}_{rsg} \subseteq rsg^{i+1}
\]

RSG(I)(rsg) = \bigcup_{1 \leq i}(\delta^i_{rsg}).

less redundancy
An improvement

\[ \delta_{rsg}^{i+1} \neq rsg^{i+1} - rsg^i \]
e.g., \((g, f) \in \delta_{rsg}^2\), not in \(rsg^2 - rsg^1\)
use \(rsg^i - rsg^{i-1}\) instead of place of \(\Delta_{rsg}^i\) in the second
“rule” of RSG’

Non linear rules : e.g., ancestor

\[
\begin{align*}
\text{anc}(x, y) & \leftarrow \text{par}(x, y) \\
\text{anc}(x, y) & \leftarrow \text{anc}(x, z), \text{anc}(z, y)
\end{align*}
\]

semi-naive evaluation

\[
\begin{align*}
\Delta_{\text{anc}}^1(x, y) & \leftarrow \text{par}(x, y) \\
\Delta_{\text{anc}}^{i+1}(x, y) & \leftarrow \Delta_{\text{anc}}^i(x, z), \text{anc}^i(z, y) \\
\Delta_{\text{anc}}^{i+1}(x, y) & \leftarrow \text{anc}^i(x, z), \Delta_{\text{anc}}^i(z, y)
\end{align*}
\]

still some redundancy : use

\[
\begin{align*}
temp^{i+1}(x, y) & \leftarrow \Delta_{\text{anc}}^i(x, z), \text{anc}^{i-1}(z, y) \\
temp^{i+1}(x, y) & \leftarrow \text{anc}^i(x, z), \Delta_{\text{anc}}^i(z, y)
\end{align*}
\]
Top-down technique: Query-Sub-Query

Program + query

\[
\begin{align*}
\text{rsg}(x, y) & \leftarrow \text{flat}(x, y) \\
\text{rsg}(x, y) & \leftarrow \text{up}(x, x1), \text{rsg}(y1, x1), \text{down}(y1, y) \\
\text{query}(y) & \leftarrow \text{rsg}(a, y)
\end{align*}
\]

Focus on relevant data
avoid deriving unnecessary tuples
A is relevant if there is a proof tree for \textit{query} in which the fact occurs

We will do that (not perfectly) using four ingredients

(a) SLD-resolution but set-at-a-time using relational algebra
(b) start with query constants and “push” constants from goals to subgoals (in the style of to pushing selections into joins)
(c) Use “sideways information passing” to pass constant binding information from one atom to the next in subgoals
(d) Use an efficient global flow-of-control strategy
Technique

adornment and adorned rules
supplementary relations and QSQ templates
the kernel of the technique
global control strategies
Adornment

\[ rsg(a, y) \]

We have a binding for the first argument of \( rsg \)

We denote it \( rsg^{bf} \)

Later on, we need \( rsg^{fb} \)

Based on the evaluation of subqueries \( (rsg^{fb}, \{\langle e \rangle, \langle f \rangle\}) \)

In general \( (R^\gamma, J) \) where

1. \( R \) is a predicate
2. \( \gamma \) an adornment
3. \( J \) provides bindings
4. completion for \( t \) in \( J \)
5. answer of the subquery : set of completions
Sideways information passing

from relational optimization

- evaluation of joins
- $P(a,v)$ join $Q(b,w,x)$ join $R(v,w,y)$
- evaluate first $P(a,v) \rightarrow$ obtains some $v$’s
- evaluate then $R(v,w,y)$ restricted by $v$’s $\rightarrow$ obtains some $w$’s
- finally evaluate $Q(b,w,x)$ restricted by $w$’s

Adorned rule

$R(x, y, z) \leftarrow R_1(x, u, v), R_2(u, w, w, z), R_3(v, w, y, a)$

a subquery involving $R^{bfb}$ with left-to-right evaluation (reorder if necessary)

$R^{bfb}(x, y, z) \leftarrow$

$R^{bff}(x, u, v), R_2^{bff}(u, w, w, z), R_3^{bff}(v, w, y, a)$

given head adornment and an ordering of body, algo for adorning
Supplementary relations and QSQ templates

Data structure to store information during evaluation

$n + 1$ supplementary relations for a rule with $n$ atoms

\[
R^{bfb}(x, y, z) \leftarrow R^{bff}_1(x, u, v), R^{bff}_2(u, w, w, z), R^{bfb}_3(v, w, y, a) \\
\uparrow \uparrow \uparrow \uparrow \\
sup_0[x, z] \quad sup_1[x, z, u, v] \quad sup_2[x, z, v, w] \quad sup_3[x, y, z]
\]

variables serve as attribute names in the supplementary relations

QSQ template for the adorned rule: $sup_0, sup_1, sup_2, sup_3$
Kernel of QSQ evaluation

input : program + query

construct adorned rules for the query :

(1) \( rsg^{bf}(x, y) \leftarrow flat(x, y) \)
(2) \( rsg^{fb}(x, y) \leftarrow flat(x, y) \)
(3) \( rsg^{bf}(x, y) \leftarrow up(x, x1), rsg^{fb}(y1, x1), down(y1, y) \)
(4) \( rsg^{fb}(x, y) \leftarrow down(y1, y), rsg^{bf}(y1, x1), up(x, x1) \)

(Note the ordering to pass bindings in (4))

construct QSQ template for each adorned rule i and the supplementary relation variables \( sup^i_j \)

construct also : for each idb R and adornment \( \gamma \)

1. the variable \( ans\_R^\gamma \) of arity \( \text{arity}(R) \)
2. the variable \( input\_R^\gamma \) of arity \( \text{bound}(R, \gamma) \)

initialization : use the query
Kinds of steps

A : from $input_R^\gamma$ to $sup_0^i$
move new tuples in $input_R^\gamma$ to 0th supplementary relations

B : from $sup_j^i$ to $sup_{j+1}^i$
Pass new tuples from one supplementary relation to the next

C : from $ans_R^\gamma$ to $sup_j^i$
Use new idb tuples (from $ans_R^\gamma$) to generate new supplementary relation tuple

D : from $sup_n^i$ to $ans_R^\gamma$
Process tuples in the final supplementary relation of a rule
Example

\[ \langle a \rangle \] into \( \text{input}_\text{rsg}^{bf} \) init
\[ \langle a \rangle \] \( \sup_0^1, \sup_0^3 \) \( A \)
\[ \langle a, e \rangle, \langle a, f \rangle \] \( \sup_1^3 \) \( B \)
\[ \langle e \rangle, \langle f \rangle \] \( \text{input}_\text{rsg}^{fb} \) \( B \)
\[ \langle g, f \rangle \] \( \text{ans}_\text{rsg}^{fb} \) B,D and 2nd rule
\[ \langle a, b \rangle \] \( \text{ans}_\text{rsg}^{bf} \) C,B,D and 3rd rule
Global control strategies

basic one : apply steps (A) through (D) until a fixpoint is reached
QSQR (query-subquery-recursive)
in step (B) : pass from supplementary relation \( sup_{i-1}^j \) across an idb predicate \( R^\gamma \) to supplementary relation \( sup_i^j \)
introduce new tuples into \( sup_i^j \) and into \( input\_R^\gamma \)
perform a recursive call to determine the \( R^\gamma \)-values corresponding to the new tuples added to \( input\_R^\gamma \), before considering the new tuples placed into \( sup_i^j \)
see algorithm in the book
Merci