Datalog

Serge Abiteboul

INRIA

5 mai 2009
Read Chapter 12 of [AHV]

Two alternative viewpoints
database vs. logic programming

\[
\begin{array}{llll}
R & S & I = \{ & R(0,1), S(1,2), \\
A & B & & S(1,1), S(1,2) \\
\hline
0 & 1 & 1 & 1 \\
1 & 2 & 1 & 2 \\
\end{array}
\]

Set of facts

\( I(R), I(S) \)

Columns are named \quad Columns are numbered
Datalog - Syntax

- **datalog rule**: $R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n)$ for $n \geq 1$
  - each variable occurring in head $u_1$
  - must also occur in body $u_2, \ldots, u_n$
- **datalog program**: finite set of datalog rules
- $adom(P, I) = adom(P) \cup adom(I)$
- **valuation** $\nu$: mapping from $\text{var}$ to $\text{dom}$
- **instantiation**: $R_1(\nu(u_1)) \leftarrow R_2(\nu(u_2)), \ldots, R_n(\nu(u_n))$
- **extensional relation**: occurs only in the body of the rules
- **intensional relation**: occurs in one head of some rule
- $\text{sch}(P) = \text{edb}(P) \cup \text{idb}(P)$
- **semantics of $P$**: maps instances over $\text{edb}(P)$ to instances over $\text{idb}(P)$
Example: metro database

<table>
<thead>
<tr>
<th>Line</th>
<th>Station</th>
<th>Next-Station</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>St-Germain</td>
<td>Odeon</td>
</tr>
<tr>
<td>4</td>
<td>Odeon</td>
<td>St-Michel</td>
</tr>
<tr>
<td>4</td>
<td>St-Michel</td>
<td>Chatelet</td>
</tr>
<tr>
<td>1</td>
<td>Chatelet</td>
<td>Louvres</td>
</tr>
<tr>
<td>1</td>
<td>Louvres</td>
<td>Palais-Royal</td>
</tr>
<tr>
<td>1</td>
<td>Palais-Royal</td>
<td>Tuileries</td>
</tr>
<tr>
<td>1</td>
<td>Tuileries</td>
<td>Concorde</td>
</tr>
<tr>
<td>9</td>
<td>Pont de Sevres</td>
<td>Billancourt</td>
</tr>
<tr>
<td>9</td>
<td>Billancourt</td>
<td>Michel-Ange</td>
</tr>
<tr>
<td>9</td>
<td>Michel-Ange</td>
<td>Iena</td>
</tr>
<tr>
<td>9</td>
<td>Iena</td>
<td>F.D. Roosevelt</td>
</tr>
<tr>
<td>9</td>
<td>F.D. Roosevelt</td>
<td>Republique</td>
</tr>
<tr>
<td>9</td>
<td>Republique</td>
<td>Voltaire</td>
</tr>
</tbody>
</table>
Example : (recursive) program \( P \)

\[
\begin{align*}
St\text{-}Reachable(x, x) & \leftarrow \\
St\text{-}Reachable(x, y) & \leftarrow St\text{-}Reachable(x, z), Links(u, z, y) \\
Li\text{-}Reachable(x, u) & \leftarrow St\text{-}Reachable(x, z), Links(u, z, y) \\
Ans\_1(y) & \leftarrow St\text{-}Reachable(Odeon, y) \\
Ans\_2(u) & \leftarrow Li\text{-}Reachable(Odeon, u) \\
Ans\_3() & \leftarrow St\text{-}Reachable(Odeon, Chatelet)
\end{align*}
\]

\[
\begin{align*}
sch(P_{metro}) & = \{ Links, St\text{-}Reachable, Li\text{-}Reachable, Ans\_1, Ans\_2, Ans\_3 \} \\
edb(P_{metro}) & = \{ Links \} \\
idb(P_{metro}) & = \{ St\text{-}Reachable, Li\text{-}Reachable, Ans\_1, Ans\_2, Ans\_3 \} \\
\nu : \nu(x) = "Odeon", \nu(y) = "Chatelet", \nu(z) = "Louvres", \nu(u) = 2
\end{align*}
\]

instantiations

\[
\begin{align*}
St\text{-}Reachable("Odeon", "Chatelet") & \leftarrow \\
St\text{-}Reachable("Odeon", "Louvres"), Links(2, "Louvres", "Chatelet") & \leftarrow \\
St\text{-}Reachable("Odeon", "Chatelet") & \leftarrow \\
St\text{-}Reachable("Odeon", "Louvres"), Links(2, "Louvres", "PaloAlto") & \leftarrow
\end{align*}
\]
Model theoretic semantics

- view \( P \) as a first-order sentence that describes the answer
- associate a formula to a rule \( r = R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n) : \)
  \[
  \forall x_1, \ldots, x_m(R_1(u_1) \leftarrow R_2(u_2) \land \ldots \land R_n(u_n))
  \]
  where \( x_1, \ldots, x_m \) are the variables occurring in the rule
- \( P = \{r_1, \ldots, r_n\}, \Sigma_P = r_1 \land \ldots \land r_n \)
- the answer is a particular model of \( \Sigma_P \)
- \( I \) over \( sch(P) \) is a model of \( \Sigma_P \) (\( I \) satisfies \( \Sigma_P \))
  if \( I \models r_1 \land \ldots \land r_n \)
- \( I \models r : \) for each \( \nu \), if \( R_2(\nu(u_2)), \ldots, R_n(\nu(u_n)) \) belong to \( I \),
  then \( R_1(\nu(u_1)) \) also belongs to \( I \)
- equivalent formulas
  \[
  \forall x_1, \ldots, x_q(\exists x_{q+1}, \ldots, x_m(R_2(u_2) \land \ldots \land R_n(u_n)) \rightarrow R_1(u_1))
  \]
  \[
  \forall x_1, \ldots, x_m(R_1(u_1) \lor \neg R_2(u_2) \lor \ldots \lor \neg R_n(u_n)).
  \]

- Horn clauses: disjunction of literals of which at most one is positive
Smallest model of $P$ containing $I$

- Intuition: these facts are sure; other facts are not
- Closed World Assumption
- A datalog program $P$, an instance $I$ over $edb(P)$
- A model of $P$ is an instance over $sch(P)$ satisfying $\Sigma_P$
- The semantics of $P$ on input $I$, denoted $P(I)$, is the minimum model of $P$ containing $I$
- Problems:
  1. Is my definition correct?
  2. How do I compute it efficiently?
### Transitive closure

\[
P(x,y) \leftarrow R(x,y) \\
P(x,y) \leftarrow R(x,z), P(z,y) \\
I = \{ R(01,), R(1,2), R(2,3) \}
\]

<table>
<thead>
<tr>
<th>(K)</th>
<th>(K')</th>
<th>(P(I))</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>1 2</td>
<td>1 2</td>
<td>1 2</td>
</tr>
<tr>
<td>2 3</td>
<td>2 3</td>
<td>2 3</td>
</tr>
<tr>
<td>0 2</td>
<td>0 2</td>
<td>0 2</td>
</tr>
<tr>
<td>1 3</td>
<td>1 3</td>
<td>1 3</td>
</tr>
<tr>
<td>0 3</td>
<td>0 3</td>
<td></td>
</tr>
<tr>
<td>6 6</td>
<td>6 6</td>
<td></td>
</tr>
</tbody>
</table>
Theorem

- The smallest instance over sch(P) satisfying $\Sigma_P$ and containing I always exists.
- Very inefficient algorithm:
  1. find the set $\mathcal{X}'$ of instances over sch(P) with values in adom(P,I), satisfying $\Sigma_P$ and containing I
  2. the result is $\bigcap \mathcal{X}'$

- proof
Fixpoint semantics

- Immediate consequence operator
- \( P \) a datalog program, \( K \) an instance over \( \text{sch}(P) \)
- A fact \( A \) is an immediate consequence for \( K \) and \( P \) if:
  1. \( A \in K(R) \) for some edb \( R \), or
  2. \( A \leftarrow A_1, \ldots, A_n \) instantiation of a rule in \( P \) and each \( A_i \in K \)

- \( T_P : \text{inst}(\text{sch}(P)) \rightarrow \text{inst}(\text{sch}(P)) \)
  \( T_P(K) = \{ \text{immediate consequences for } K \text{ and } P \} \)

- example
Some facts

- **Fact 1**: \( T_P \) is monotone
  \[ \forall I, J, I \subseteq J \implies T_P(I) \subseteq T_P(J) \]

- **Definition**: \( K \) is a fixpoint of \( T_P \) if \( T_P(K) = K \)

- **Fact 2**: \( K \) over \( sch(P) \) is a model of \( \Sigma_P \) iff \( T_P(K) \subseteq K \)

- **Fact 3**: Each fixpoint of \( T_P \) is a model of \( \Sigma_P \) the converse does not necessarily hold.

- **Fact 4**: For each \( P \) and \( I \),
  \( T_P \) has a minimum fixpoint containing \( I \), which equals \( P(I) \).
Construction

- Compute $T_P(I)$, $T_P^2(I)$, $T_P^3(I)$, etc.
- $I \subseteq T_P(I) \subseteq T_P^2(I) \subseteq T_P^3(I) \ldots \subseteq B(P, I)$
- $\{T_P^i(I)\}_i$ reaches a fixpoint after at most $N$ steps: $T_P(T_P^N(I)) = T_P^N(I)$.
- We denote this fixpoint $T_P^\omega(I)$
- Theorem: $P$ be a datalog program, $I$ an instance over $edb(P)$, $T_P^\omega(I) = P(I)$
Example

- $P_{TC}$:
  \[ T(x, y) \leftarrow G(x, y) \]
  \[ T(x, y) \leftarrow G(x, z), T(z, y). \]
  \[ I = \{ G(1, 2), G(2, 3), G(3, 4), G(4, 5) \} \]

- This leads to:
  \[
  T_{P_{TC}}(I) = I \cup \{ T(1, 2), T(2, 3), T(3, 4), T(4, 5) \}
  \]
  \[
  T_{P_{TC}}^2(I) = T_{P_{TC}}(I) \cup \{ T(1, 3), T(2, 4), T(3, 5) \}
  \]
  \[
  T_{P_{TC}}^3(I) = T_{P_{TC}}^2(I) \cup \{ T(1, 4), T(2, 5) \}
  \]
  \[
  T_{P_{TC}}^4(I) = T_{P_{TC}}^3(I) \cup \{ T(1, 5) \}
  \]
  \[
  T_{P_{TC}}^5(I) = T_{P_{TC}}^4(I).
  \]

- $T_{P_{TC}}^\omega(I) = T_{P_{TC}}^4(I)$

- Proof of theorem
Stage

- The smallest integer $i$ such that $T_P^i(I) = T_P^\omega(I)$ is called the stage for $P$ and $I$

EVALUATION

- Algorithm by example

- Transitive closure

$$T := G;$$
while $q(T) \neq T$ do $T := q(T);$ 

where

$$q(T) = G \cup \pi_{AB}(\delta_{B\rightarrow C}(G) \Join \delta_{A\rightarrow C}(T)).$$

($\delta$ is the renaming operation.)
Proof-theoretic approach

\[ S(x_1, x_3) \leftarrow T(x_1, x_2), R(x_2, a, x_3). \]

- \[ T(x_1, x_4) \leftarrow R(x_1, a, x_2), R(x_2, b, x_3), T(x_3, x_4). \]
- \[ T(x_1, x_3) \leftarrow R(x_1, a, x_2), R(x_2, a, x_3). \]
- \{ R(1, a, 2), R(2, b, 3), R(3, a, 4), R(4, a, 5), R(5, a, 6) \}
A technique for deriving proofs: SLD resolution

Warm-up: only ground rules and ground facts

1. $S(1,6) \leftarrow T(1,5), R(5,a,6)$
2. $T(1,5) \leftarrow R(1,a,2), R(2,b,3), T(3,5)$
3. $T(3,5) \leftarrow R(3,a,4), R(4,a,5)$
4. $R(1,a,2) \leftarrow$
5. $R(2,b,3) \leftarrow$
6. $R(3,a,4) \leftarrow$
7. $R(4,a,5) \leftarrow$
8. $R(5,a,6) \leftarrow$
Technique: refutation

(1) \[ S(1, 6) \leftarrow T(1, 5), R(5, a, 6) \]
(2) \[ T(1, 5) \leftarrow R(1, a, 2), R(2, b, 3), T(3, 5) \]
(3) \[ T(3, 5) \leftarrow R(3, a, 4), R(4, a, 5) \]
(4) \[ R(1, a, 2) \leftarrow \]
(5) \[ R(2, b, 3) \leftarrow \]
(6) \[ R(3, a, 4) \leftarrow \]
(7) \[ R(4, a, 5) \leftarrow \]
(8) \[ R(5, a, 6) \leftarrow \]

Goal

\[
\neg S(1, 6)
\]

\[
\Rightarrow \neg T(1, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg R(1, a, 2) \lor \neg R(2, b, 3) \lor \neg T(3, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg R(2, b, 3) \lor \neg T(3, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg T(3, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg R(3, a, 4) \lor \neg R(4, a, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg R(4, a, 5) \lor \neg R(5, a, 6)
\]

\[
\Rightarrow \neg R(5, a, 6)
\]

\[
\Rightarrow false
\]

Rule used

(1)
SLD resolution

- we start with a program (includes the db facts)

1. \( S(x_1, x_3) \leftarrow T(x_1, x_2), R(x_2, a, x_3) \)
2. \( T(x_1, x_4) \leftarrow R(x_1, a, x_2), R(x_2, b, x_3), T(x_3, x_4) \)
3. \( T(x_1, x_3) \leftarrow R(x_1, a, x_2), R(x_2, a, x_3) \)
4. \( R(1, a, 2) \leftarrow \)
5. \( R(2, b, 3) \leftarrow \)
6. \( R(3, a, 4) \leftarrow \)
7. \( R(4, a, 5) \leftarrow \)
8. \( R(5, a, 6) \leftarrow \)

- use unification: a unifier for two atoms A, B is a substitution \( \theta \) such that \( \theta A = \theta B \)
- most general unifier \( \theta \): for each unifier \( \nu \), there is a substitution \( \nu \) such that \( \nu = \theta \circ \nu' \)
- one step of resolution
goal $\leftarrow A_1, \ldots, A_i, \ldots, A_n$

rule $B_1 \leftarrow B_2, \ldots, B_m$

$\theta$ mgu of $A_i$ and $B_1$

new goal $\leftarrow \theta(A_1), \ldots, \theta(A_{i-1}), \theta(B_2), \ldots, \theta(B_m)$,
$\theta(A_{i+1}), \ldots, \theta(A_n)$
SLD-resolution continued

- This also provides binding for the variables of the original goal
- \( \leftarrow T(x, y) \) provides all the pairs \((x, y)\) such that \( \neg T(x, y) \) is false, i.e., \( T(x, y) \) holds
- soundness and completeness
  (a,b) is in the answer to \( \leftarrow T(x, y) \) iff (a,b) is in \( P(I)(T) \)
- SLD TREES: top-down technique
- nondeterminism
  1. which atom to select in the current goal
  2. which rule to use
- Use a selection rule to select goal
  Prolog selects the leftmost atom of the goal
- Once an atom has been selected, try all possible rules
Static program analysis

- satisfiability of $T$
  is there an extensional database $I$ such that $P(I)(T)$ is nonempty
- containment for $T$
  for each extensional database $I$,
  $P(I)(T)$ is included in $P'(I)(T)$
- optimization
- boundedness
  the fixpoint is reached after a bounded number of steps
  more optimization
- containment and boundedness are undecidable
- optimization will be difficult
  heuristics
Merci