Datalog

Serge Abiteboul

INRIA

5 mai 2009

Datalog

- Read Chapter 12 of [AHV]
- Two alternative viewpoints
  database vs. logic programming

```
R       S              I = { R(0,1), S(1,2),
A  B    A  C                 S(1,1), S(1,2) }
-----   -----
0  1    1  1
1  2    1  2           Set of facts

I(R), I(S)

Columns are named  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
- $adom(P, \mathbf{I}) = adom(P) \cup adom(\mathbf{I})$
- valuation ν : mapping fro **var** to **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
- sch(P) $=$ edb(P) $\cup$ idb(P)
- semantics of P : maps instances over $edb(P)$ to instances over $idb(P)$

Example : metro database

Links	Line	Station	Next-Station
	4	*St-Germain*	*Odeon*
	4	*Odeon*	*St-Michel*
	4	*St-Michel*	*Chatelet*
	1	*Chatelet*	*Louvres*
	1	*Louvres*	*Palais-Royal*
	1	*Palais-Royal*	*Tuileries*
	1	*Tuileries*	*Concorde*
	9	*Pont de Sevres*	*Billancourt*
	9	*Billancourt*	*Michel-Ange*
	9	*Michel-Ange*	*Iena*
	9	*Iena*	*F.D. Roosevelt*
	9	*F.D. Roosevelt*	*Republique*
	9	*Republique*	*Voltaire*

Example : (recursive) program P

$$
\begin{array}{lcl}
St\_Reachable(x,x) & \leftarrow & \\
St\_Reachable(x,y) & \leftarrow & St\_Reachable(x,z), Links(u,z,y) \\
Li\_Reachable(x,u) & \leftarrow & St\_Reachable(x,z), Links(u,z,y) \\
Ans\_1(y) & \leftarrow & St\_Reachable(Odeon,y) \\
Ans\_2(u) & \leftarrow & Li\_Reachable(Odeon,u) \\
Ans\_3() & \leftarrow & St\_Reachable(Odeon, Chatelet)
\end{array}
$$

$sch(P_{metro}) = \{Links,\ St\_Reachable,\ Li\_Reachable,\ Ans\_1,\ Ans\_2,\ Ans\_3\ \}$
$edb(P_{metro}) = \{Links\}$
$idb(P_{metro}) = \{St\_Reachable,\ Li\_Reachable,\ Ans\_1,\ Ans\_2,\ Ans\_3\ \}$
$\nu : \nu(x) = "Odeon",\ \nu(y) = "Chatelet",\ \nu(z) = "Louvres",\ \nu(u) = 2$
instantiations
$St\_Reachable("Odeon","Chatelet") \leftarrow$
$St\_Reachable("Odeon","Louvres"), Links(2,"Louvres","Chatelet")$
$St\_Reachable("Odeon","Chatelet") \leftarrow$
$St\_Reachable("Odeon","Louvres"), Links(2,"Louvres","PaloAlto")$

Model theoretic semantics

- view P as a first-order sentence that describes the answer
- associate a formula to a rule r $= R_1(u_1) \leftarrow R_2(u_2), \ldots, R_n(u_n)$:

$$\forall x_1, \ldots, x_m (R_1(u_1) \leftarrow R_2(u_2) \wedge \ldots \wedge R_n(u_n))$$

  where $x_1, \ldots, x_m$ are the variables occurring in the rule
- $P = \{r_1, ..., r_n\}$, $\Sigma_P = r_1 \wedge ... \wedge r_n$
- the answer is a particular model of Σ_P
- $\mathbf{I}$ over $sch(P)$ is a model of Σ_P ($\mathbf{I}$ satisfies Σ_P) if $\mathbf{I} \models r_1 \wedge ... \wedge r_n$
- $\mathbf{I} \models r$: for each ν, if $R_2(\nu(u_2)), \ldots, R_n(\nu(u_n))$ belong to $\mathbf{I}$, then $R_1(\nu(u_1))$ also belongs to $\mathbf{I}$
- equivalent formulas

$$\forall x_1, \ldots, x_q (\exists x_{q+1}, \ldots, x_m (R_2(u_2) \wedge \ldots \wedge R_n(u_n)) \rightarrow R_1(u_1))$$

$$\forall x_1, \ldots, x_m (R_1(u_1) \vee \neg R_2(u_2) \vee \ldots \vee \neg R_n(u_n)).$$

- 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 $\mathbf{I}$ over $edb(P)$
- A model of P is an instance over $sch(P)$ satisfying Σ_P
- The *semantics* of P on input $\mathbf{I}$, denoted $P(\mathbf{I})$, is the minimum model of P containing $\mathbf{I}$
- Problems :
  1. Is my definition correct ?
  2. how do I compute it efficiently ?

example

- Transitive closure
  P(x,y) ← R(x,y)
  P(x,y) ← R(x,z), P(z,y)
  I = { R(01,), R(1,2), R(2,3) }

```
K                    K'                   P(I)
----------           ----------           ----------
R        P           R        P           R        P
----     ----        ----     ----        ----     ----
0 1      0 1         0 1      0 1         0 1      0 1
1 2      1 2         1 2      1 2         1 2      1 2
2 3      2 3         2 3      2 3         2 3      2 3
         0 2                  0 2                  0 2
         1 3                  1 3                  1 3
         0 3                                       0 3
         6 6
```

Theorem

- The smallest instance over sch(P) satisfying Σ_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 Σ_P and containing I
  2. the result is $\bigcap \mathcal{X}'$
- proof

Fixpoint semantics

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

Some facts

- Fact1 : T_P is monotone
  $\forall$ **I**, **J**, $\mathbf{I} \subseteq \mathbf{J}$ implies $T_P(\mathbf{I}) \subseteq T_P(\mathbf{J})$
- Def : **K** is a *fixpoint* of T_P if $T_P(\mathbf{K}) = \mathbf{K}$
- Fact2 : **K** over $sch(P)$ is a model of Σ_P iff $T_P(\mathbf{K}) \subseteq \mathbf{K}$
- Fact3 : Each fixpoint of T_P is a model of Σ_P
  the converse does not necessarily hold.
- Fact4 : For each P and **I**,
  T_P has a minimum fixpoint containing **I**, which equals $P(\mathbf{I})$.

Construction

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

Example

- P_{TC} : $\begin{array}{rcl} T(x,y) & \leftarrow & G(x,y) \\ T(x,y) & \leftarrow & G(x,z), T(z,y). \end{array}$
  $\mathbf{I} = \{G(1,2), G(2,3), G(3,4), G(4,5)\}$
- This leads to :

$$\begin{array}{rcl} T_{P_{TC}}(I) & = & I \cup \{T(1,2), T(2,3), T(3,4), T(4,5)\} \\ T^2_{P_{TC}}(I) & = & T_{P_{TC}}(I) \cup \{T(1,3), T(2,4), T(3,5)\} \\ T^3_{P_{TC}}(I) & = & T^2_{P_{TC}}(I) \cup \{T(1,4), T(2,5)\} \\ T^4_{P_{TC}}(I) & = & T^3_{P_{TC}}(I) \cup \{T(1,5)\} \\ T^5_{P_{TC}}(I) & = & T^4_{P_{TC}}(I). \end{array}$$

- $T^{\omega}_{P_{TC}}(I) = T^4_{P_{TC}}(I)$
- proof of theorem

Stage

- The smallest integer i such that $T_P^i(\mathbf{I}) = T_P^\omega(\mathbf{I})$ is called the stage for P and $\mathbf{I}$
- EVALUATION
- Algorithm by example
- Transitive closure

$$\begin{array}{l} T \;:=\; G; \\ \text{while } q(T) \neq T \text{ do } T := q(T); \end{array}$$

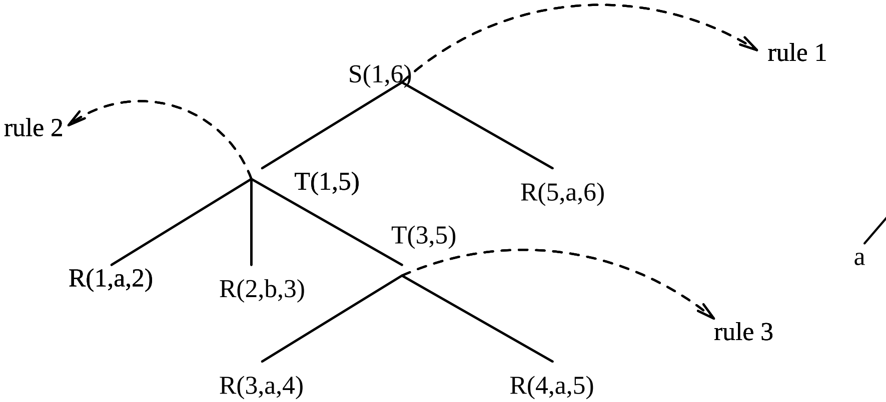
where

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

- (δ is the renaming operation.)

Proof-theoretic approach

- $$\begin{array}{rcl} S(x_1, x_3) & \leftarrow & T(x_1, x_2), R(x_2, a, x_3). \\ T(x_1, x_4) & \leftarrow & R(x_1, a, x_2), R(x_2, b, x_3), T(x_3, x_4). \\ T(x_1, x_3) & \leftarrow & R(x_1, a, x_2), R(x_2, a, x_3). \end{array}$$
- $\{R(1, a, 2), R(2, b, 3), R(3, a, 4), R(4, a, 5), R(5, a, 6)\}$



A technique for deriving proofs : SLD resolution

- Warm-up : only ground rules and ground facts

$$
\begin{array}{llll}
(1) & S(1,6) & \leftarrow & T(1,5), R(5,a,6) \\
(2) & T(1,5) & \leftarrow & R(1,a,2), R(2,b,3), T(3,5) \\
(3) & T(3,5) & \leftarrow & R(3,a,4), R(4,a,5) \\
(4) & R(1,a,2) & \leftarrow & \\
(5) & R(2,b,3) & \leftarrow & \\
(6) & R(3,a,4) & \leftarrow & \\
(7) & R(4,a,5) & \leftarrow & \\
(8) & R(5,a,6) & \leftarrow &
\end{array}
$$

<-- S(1,6)	S(1,6) <-- T(1,5), R(5,a,6)
<-- T(1,5), R(5,a,6)	T(1,5) <-- R(1,a,2), R(2,b,3), T(3,5)
<-- R(1,a,2), R(2,b,3), T(3,5), R(5,a,6)	R(1,a,2) <--
<-- R(2,b,3), T(3,5), R(5,a,6)	R(2,b,3) <--
<-- T(3,5), R(5,a,6)	T(3,5) <-- R(3,a,4), R(4,a,5)
<-- R(3,a,4), R(4,a,5), R(5,a,6)	R(3,a,4) <--
<-- R(4,a,5), R(5,a,6)	R(4,a,5) <--
<-- R(5,a,6)	R(5,a,6) <--
<--	

Technique : refutation

$$
\begin{array}{llll}
(1) & S(1,6) & \leftarrow & T(1,5), R(5,a,6) \\
(2) & T(1,5) & \leftarrow & R(1,a,2), R(2,b,3), T(3,5) \\
(3) & T(3,5) & \leftarrow & R(3,a,4), R(4,a,5) \\
(4) & R(1,a,2) & \leftarrow & \\
(5) & R(2,b,3) & \leftarrow & \\
(6) & R(3,a,4) & \leftarrow & \\
(7) & R(4,a,5) & \leftarrow & \\
(8) & R(5,a,6) & \leftarrow &
\end{array}
$$

	Goal	Rule used
	$\neg S(1,6)$	(1)
$\Rightarrow$	$\neg T(1,5) \vee \neg R(5,a,6)$	(2)
$\Rightarrow$	$\neg R(1,a,2) \vee \neg R(2,b,3) \vee \neg T(3,5) \vee \neg R(5,a,6)$	(4)
$\Rightarrow$	$\neg R(2,b,3) \vee \neg T(3,5) \vee \neg R(5,a,6)$	(5)
$\Rightarrow$	$\neg T(3,5) \vee \neg R(5,a,6)$	(3)
$\Rightarrow$	$\neg R(3,a,4) \vee \neg R(4,a,5) \vee \neg R(5,a,6)$	(6)
$\Rightarrow$	$\neg R(4,a,5) \vee \neg R(5,a,6)$	(7)
$\Rightarrow$	$\neg R(5,a,6)$	(8)
$\Rightarrow$	*false*	

SLD resolution

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

$$
\begin{array}{llcl}
(1) & S(x_1, x_3) & \leftarrow & T(x_1, x_2), R(x_2, a, x_3) \\
(2) & T(x_1, x_4) & \leftarrow & R(x_1, a, x_2), R(x_2, b, x_3), T(x_3, x_4) \\
(3) & T(x_1, x_3) & \leftarrow & R(x_1, a, x_2), R(x_2, a, x_3) \\
(4) & R(1, a, 2) & \leftarrow & \\
(5) & R(2, b, 3) & \leftarrow & \\
(6) & R(3, a, 4) & \leftarrow & \\
(7) & R(4, a, 5) & \leftarrow & \\
(8) & R(5, a, 6) & \leftarrow &
\end{array}
$$

- use unification : a unifier for two atoms A, B is a substitution θ such that $\theta A = \theta B$
- most general unifier θ : for each unifier ν, there is a substitution ν such that $\nu = \theta \circ \nu'$
- one step of resolution

$$
\begin{array}{ll}
\mathit{goal} & \leftarrow A_1, \ldots, A_i, \ldots, A_n \\
\mathit{rule} & B_1 \leftarrow B_2, \ldots, B_m \\
\theta & \text{mgu of } A_i \text{ and } B_1 \\
\mathit{new\ goal} & \leftarrow \theta(A_1), \ldots, \theta(A_{i-1}), \theta(B_2), \ldots, \theta(B_m), \\
 & \theta(A_{i+1}), \ldots, \theta(A_n)
\end{array}
$$

<-- S(1,x)

S(x1,x3) <-- T(x1,x2), R(x2,a,x3)

<-- T(1,x2), R(x2,a,x)

T(x1,x4) <-- R(x1,a,x2), R(x2,b,x3), T(x3,x4)

<-- R(1,a,y2), R(y2,b,x3), T(x3,x2), R(x2,a,x)

R(1,a,2) <--

<-- R(2,b,x3), T(x3,x2), R(x2,a,x)

R(2,b,3) <--

<-- T(3,x2), R(x2,a,x)

T(x1,x3) <-- R(x1,a,x2), R(x2,a,x3)

<-- R(3,a,z2), R(z2,a,x2), R(x2,a,x)

R(3,a,4) <--

<-- R(4,a,x2), R(x2,a,x)

R(4,a,5) <--

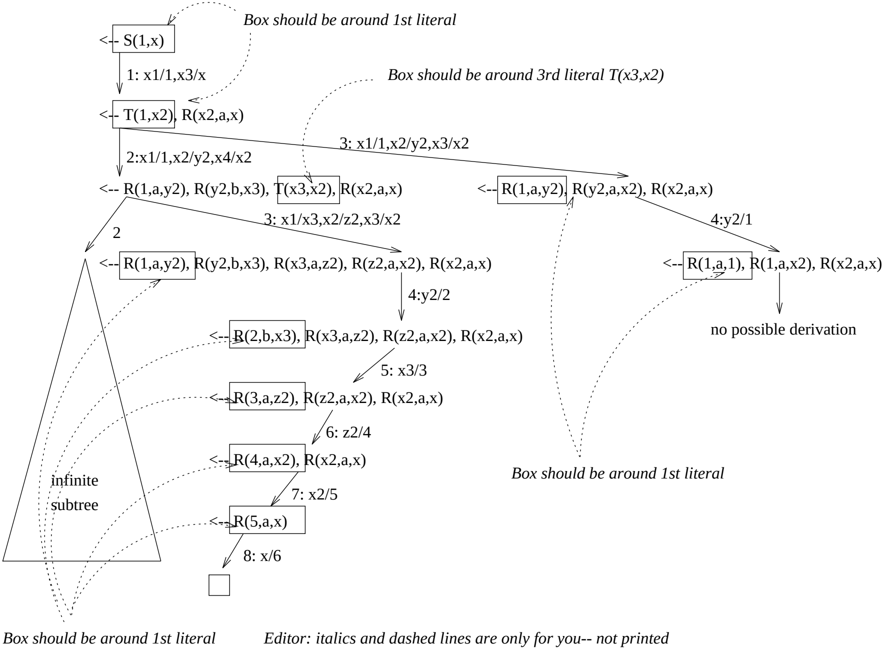
<-- R(5,a,x)

R(5,a,6) <--

<--

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



Editor: italics and dashed lines are only for you-- not printed

Editor: please put number after x,y,z as indexes if possible

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