Programming Languages:
Logic Paradigm

Onur Tolga Şehitoğlu

Computer Engineering, METU

8 May 2008
Logic Programming Paradigm

2 Prolog basics

3 Prolog Terms

4 Unification

5 Declarations

6 Procedural Interpretation
Logic Programming Paradigm

- Based on logic and declarative programming
- 60’s and early 70’s
- Prolog (*Programming in logic*, 1972) is the most well known representative of the paradigm.
- Prolog is based on Horn clauses and SLD resolution
- Mostly developed in fifth generation computer systems project
- Specially designed for theorem proof and artificial intelligence but allows general purpose computation.
- Some other languages in paradigm: ALF, Frill, Gödel, Mercury, Oz, Ciao, \lambda Prolog, datalog, and CLP languages
Constraint Logic Programming

- Clause: disjunction of universally quantified literals,
  \[ \forall (L_1 \lor L_2 \lor ... \lor L_n) \]

- A logic program clause is a clause with exactly one positive literal
  \[ \forall (A \lor \neg A_1 \lor \neg A_2... \lor \neg A_n) \equiv \forall (A \iff A_1 \land A_2... \land A_n) \]

- A goal clause: no positive literal
  \[ \forall (\neg A_1 \lor \neg A_2... \lor \neg A_n) \]

- Proof by refutation, try to unsatisfy the clauses with a goal clause \( G \). Find \( \exists (G) \).

- Linear resolution for definite programs with constraints and selected atom.
What does Prolog look like?

father(ahmet, ayse).
father(hasan, ahmet).
mother(fatma, ayse).
mother(hatice, fatma).
parent(X, Y) :- father(X, Y).
parent(X, Y) :- mother(X, Y).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
- CLP on first order terms. (Horn clauses).
- **Unification.** Bidirectional.
- **Backtracking.** Proof search based on trial of all matching clauses.
Prolog Terms

- **Atoms:**
  1. Strings with starting with a small letter and consist of `a-zA-Z0-9`:
     - a  aADAM  a1_2
  2. Strings consisting of only punctuation:
     - *  ***  .+  .<>.  
  3. Any string enclosed in single quotes (like an arbitrary string):
     - 'ADAM'  'Onur Sehitoglu'  '2*4<6'

- **Numbers**
  - 1234  12.32  12.23e-10
Variables:

1. Strings with starting with a capital letter or _ and consist of
   
   \[a\text{-}zA\text{-}Z0\text{-}9]\*

   Adam _adam A093

2. _ is the universal match symbol. Not variable

Structures:

- starts with an atom head
- has one or more arguments (any term) enclosed in paranthesis, separated by comma
- structure head cannot be a variable or anything other than atom.

\[
a(b) \text{ a(b,c) a(b,c,d) } ++(12) +(*)(1,a(b)) \text{ 'hello_world'(1,2)}
\]

\[
\checkmark \\
X(b) 4(b,c) a() ++() _(3) \times
\]

- some structures defined as infix:

\[
+(1,2) \equiv 1+2 \text{ , } -(a,b,c,d) \equiv a \text{ : - } b,c,d
\]

\[
is(X,(Y,1)) \equiv X \text{ is } X + 1
\]
Syntactic Sugars

- Prolog interpreter automatically maps some easy to read syntax into its actual structure.

- **List:** \([a, b, c] \equiv \text{(a, (b, (c, []))} \]

- **Head and Tail:** \([H|T] \equiv \text{(H, T)} \]

- **String:** "ABC" \(\equiv [65, 66, 67] \) (ascii integer values)

- use `display(Term)` to see actual structure of the term.
Unification

Bi-directional (both actual and formal argument can be instantiated)

1. if $S$ and $T$ are atoms or number, unification successful only if $S = T$

2. if $S$ is a variable, $S$ is instantiated as $T$, if it is compatible with current constraint store ($S$ is instantiated to another term, they are unified)

3. if $S$ and $T$ are structures, successful if:
   - head of $S = $ head of $T$
   - they have same arity
   - unification of all corresponding terms are successful
$A$: list of atoms, $N$: list of numbers, $V$: list of variables, $S$: list of structures, $\mathcal{P}$ current constraint store

$s \in S$, $arity(s)$: number of arguments of structure,

$s \in S$, $head(s)$: head atom of the structure,

$s \in S$, $arg_i(s)$: $i^{th}$ argument term of the structure,

$p \models \mathcal{P}$: $p$ is consistent with current constraint store.

\[
S \equiv T; \mathcal{P} = \\
(S, T \in A \lor S, T \in N) \land S = T \rightarrow true; \mathcal{P} \\
S \in V \land S \equiv T \models \mathcal{P} \rightarrow true; S \equiv T \land \mathcal{P} \\
T \in V \land S \equiv T \models \mathcal{P} \rightarrow true; S \equiv T \land \mathcal{P} \\
S, T \in S \land head(S) = head(T) \land arity(S) = arity(T) \rightarrow \\
\forall i, arg_i(S) \equiv arg_i(T); \mathcal{P}
\]
Unification Examples

- $X = a \rightarrow \sqrt{\text{with } X = a}$
- $a(X, 3) = a(X, 3, 2) \rightarrow \times$
- $a(X, 3) = b(X, 3) \rightarrow \times$
- $a(X, 3) = a(3, X) \rightarrow \sqrt{\text{with } X = 3}$
- $a(X, 3) = a(4, X) \rightarrow \times$
- $a(X, b(c, d(e, f))) = a(b(c, Y), X) \rightarrow X = b(c, d(e, f)), Y = d(e, f)$
Declarations

Two types of clauses:

- **p1**(arg1, arg2, ...) :- p2(args,...) , p3(args,...).
  means if p2 and p3 true, then p1 is true. There can be arbitrary number of (conjunction of) predicates at right hand side.

- **p**(arg1, arg2, ...).
  sometimes called a fact. It is equivalent to:
  p(arg1, arg2, ...) :- true.

- **p**(args) :- q(args) ; s(args).
  Is disjunction of predicates. q or s implies p. Equivalent to:
  p(args) :- q(args).
  p(args) :- s(args).

- A prolog program is just a group of such clauses.
List Examples

% list membership
memb(X, [X|Rest]) .
memb(X, [_|Rest]) :- memb(X, Rest).

% concatenation
conc([], L, L).
conc([X|R], L, [X|R_and_L]) :- conc(R, L, R_and_L).

% second list starts with first list
prefixof([], _).
prefixof([X|Rx], [X|Ry]) :- prefixof(Rx, Ry).

% second list contains first list
sublist(L1, L2) :- prefixof(L1, L2).
sublist(L, [_|R]) :- sublist(L, R).
- For goal clause all matching head clauses (LHS of clauses) are kept as backtracking points (like a junction in maze search).
- Starts from first match.
- To prove head predicate, RHS predicates need to be proved recursively.
- If all RHS predicates are proven, head predicate is proven.
- When fails, prolog goes back to last backtracking point and tries next choice.
- When no backtracking point is left, goal clause fails.
- All predicate matches go through unification so goal clause variables can be instantiated.
Arithmetics and operators

- **X = 3+1** is not an arithmetic expression!
- Operators (*is*) force arithmetic expressions to be evaluated
- All variables of the operations needs to be instantiated
  - **12 is 3+X** does not work!
- Comparison operators force LHS and RHS to be evaluated:
  - \(X > Y, \ X < Y, \ X >= Y, \ X <= Y, \ X =: = Y, \ X == Y\)
- *is* operator forces RHS to be evaluated: \(X \ is \ Y+3*Y\) \(Y\) needs to have a numerical value when search hits this expression.
- Note that \(X \ is \ X+1\) is never successful in Prolog. Variables are instantiated once.
Greatest Common Divisor (Euler’s)

\[
gcd(m, n) = \begin{cases} 
gcd(n, m - n) & \text{if } n < m \\
gcd(n, m) & \text{if } m < n 
\end{cases}
\]

\[
gcd(X, X, X) \\
gcd(X, Y, D) :- X < Y, \ Y1 \text{ is } Y-X, \ gcd(X, Y1, D). \\
gcd(X, Y, D) :- Y < X, \ gcd(Y, X, D).
\]