Prolog

- the idea of logic programming
- Prolog and Logic
- Prolog and Databases
- how it works
- a limitation
- data structures in Prolog
The Essence of Prolog

- Prolog is a logic programming language (PROgramming in LOGic)
- It is also declarative and interactive
  - Declare facts and rules
  - Ask questions; Prolog answers.
- Deductive Inference
Prolog and Logic

- Fact examples:

  \[
  \text{parent}(\text{mary}, \text{john}). \\
  \text{female}(\text{mary}). \\
  \text{parent}(\text{ann}, \text{mary}).
  \]

- Rule examples:

  \[
  \text{mother}(X, Y) \leftarrow \text{parent}(X, Y), \text{female}(X). \\
  \text{grandparent}(X, Z) \leftarrow \text{parent}(X, Y), \text{parent}(Y, Z).
  \]

- Compare to Logic:

  \[
  \neg X \cdot \neg Y \cdot \neg Z: \neg X: \neg Y: \neg Z: \\
  \text{parent}(X, Y) \Rightarrow \text{parent}(Y, Z) \Rightarrow \text{grandparent}(X, Z)
  \]
How to Read a Rule

\[
\text{grandparent}(X, Z) :\quad \text{parent}(X, Y), \\
\quad \text{parent}(Y, Z).
\]

To prove that \( X \) is the grandparent of \( Z \) (\( \therefore \rightarrow \)):

prove that \( X \) is a parent of some \( Y \) and (\( \therefore \))

(also) that the same \( Y \) is a parent of \( Z \) (\( \therefore \))
One fact relates to many questions.

\[
\text{parent}(\text{mary}, \text{john}).
\]

• "Is Mary a parent of John?"
  \[
  |?- \text{parent}(\text{mary}, \text{john}).
  \text{yes}.
  \]

• "Who is Mary a parent of?"
  \[
  |?- \text{parent}(\text{mary}, X).
  X = \text{john}.
  \]

• "Is Mary a parent (of anyone)?"
  \[
  |?- \text{parent}(\text{mary}, _).
  \text{yes}.
  \]
One fact relates to many questions (cont.)

\[
\text{parent}(\text{mary, john}).
\]

- “Who is a parent of John?”
- “Does John have (anyone as) a parent?”
- “Who is a parent of whom?”
  \[
  | ?- \text{parent}(X,Y).
  \]
  \[
  X = \text{mary}, \ Y = \text{john}.
  \]
- “Is anyone a parent of anyone?”
Other Forms of Interaction

Facts:

parent(mary,john).
female(mary).
parent(ann,mary).

Rules:

mother(X,Y) :- parent(X,Y), female(X).

grandparent(X,Z) :- parent(X,Y), parent(Y,Z).

Queries:

| ?- parent(X,Y), parent(Y,Z).
  X=ann, Y=mary, Z = john.

| ?- parent(X,_), parent(_,Z).
  X=ann, Z=john.

| ?- grandparent(X,Y).
  X=ann, Y=john ;
  no

| ?- parent(john,ann).
  no
Databases and Prolog

• a database relation is a relation.

• select:
  empinfo(X,Y,unit23).
  empinfo(X,Y,Z), Y<1995.

• project:
  year(Y) :- empinfo(_,Y,_).
  year_unit(Y,Z) :- empinfo(_,Y,Z).

• join:
  empplus(W,X,Y,Z) :-
    empinfo(W,X,Y),
    manages(Y,Z).
Two-Clause Rules

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

different(X,X) :- !,fail.
different(_,_).
The General Form of Prolog Rules

• A rule’s body can have any number of propositions.

\[
P : - Q_1, Q_2, \ldots, Q_n.
\]

head

body

• The comma is the \(\land\) of logic.

• The logic version of this is called a Horn clause:

\[
Q_1 \land Q_2 \land \ldots \land Q_n \land P
\]

• Prolog also allows semicolons, meaning

\[
P : - Q_1; Q_2; \ldots; Q_n.
\]

• ... which is equivalent to the n-clause rule

\[
P : - Q_1.
P : - Q_2.
\ldots
P : - Q_n.
\]
What about AND \& OR
on the Left?

• To get the effect of $\land$ on the left, use

\[
P_1 \leftarrow Q.
\]
\[
P_2 \leftarrow Q.
\]
\ldots
\[
P_n \leftarrow Q.
\]

• But the effect of $\lor$ on the left is \textit{impossible}.

• That is, there are \textit{no rules} corresponding to

\[
Q_1 \lor Q_2 \lor \ldots \lor Q_n \lor P_1 \lor P_2
\]

and \textit{no facts} of the form $P_1 \lor P_2$.

• This is a \textit{limitation} on Prolog and was a deliberate choice - for the sake of efficiency.
How Prolog Works

- scope
- unification
- recursion

\[
\text{ancestor}(\text{ann}, \text{john})
\]

\[
\text{parent}(\text{ann}, \text{john}) \quad \text{fail}
\]

\[
\text{parent}(\text{ann}, Y) \quad \Box \quad \text{ancestor}(Y, \text{john})
\]

\[
Y=\text{mary}
\]

\[
\text{ancestor}(\text{mary}, \text{john})
\]

\[
\text{parent}(\text{mary}, \text{john}) \quad \text{succeed}
\]

\[
\text{ancestor}(X, Z) \leftarrow \text{parent}(X, Z).
\]

\[
\text{ancestor}(X, Z) \leftarrow \text{parent}(X, Y), \text{ancestor}(Y, Z).
\]
Structured Objects in Prolog
resemble structures or records.

Fact:

\[
\text{person(nm(john, smith), date(december, 3, 1995))}.
\]

Questions, Queries and Prolog responses:

English: "Who was born in 1995?"
Query: \(?- \text{person(P, date(\_\_\_, \_\_\_, 1995))}.
Response: P = nm(john, smith)

English: "Who was born in what month of 1970?"
Query: \(?- \text{person(nm(\_\_, L), date(M, \_\_\_, 1970))}.
Response: L = Jones, M = April
LISTS in Prolog

- Notation

  The *empty* list  \[ \]  

  3-element list  \[a,b,c\]  

  List of lists  \[[\],[a],[b,c],[d,e,f,g]\]  

  Head & Tail  \[[H|T] = [a,b,c]\].

  gives $H = a,$

  $T = [b,c]$  

- Membership (a 2-clause rule)

  \[
  \text{member}(X,[X|\text{Tail}]).
  \text{member}(X,[\text{Head}|\text{Tail}]) :- \text{member}(X,\text{Tail}).
  \]

  \[\text{?- member(c,[a,b,c,d]).}\]

  yes
Concatenation

append([],L,L).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).

?- append([a,b],[c,d],Result).
   Result = [a,b,c,d].

?- append(First, [g,h], [d,e,f,g,h]).
   First = [d,e,f].
More Rules for Lists

Push  
push(X,L,[X|L]).

Last element  
last(E,L) :- append(_, [E], L).

Delete  
delete(X,[X|Tail],Tail).

delete(X,[Y|Tail],[Y|Tail1]) :-
    delete(X,Tail,Tail1).
Even and Odd

evenlength([]).
evenlength([_|X]) :- oddlength(X).
oddlength([_|X]) :- evenlength(X).
Reverse and Palindrome

reverse([ ], [ ]).
reverse([H | T], Answer) :- reverse(T, R),
append(R, [H], Answer).

palindrome(X) :- reverse(X, X).