Logic Programming with Prolog

Prolog programs are constructed from terms:

**Constants** can be either atoms or numbers:
- Atoms: Strings of characters starting with a lower-case letter or enclosed in apostrophes.
- Numbers: Strings of digits with or without a decimal point and a minus sign.

**Variables** are strings of characters beginning with an upper-case letter or an underscore.

**Structures** consist of a functor or function symbol (looks like an atom), followed by a list of terms inside parentheses, separated by commas.

Declarative meaning:
“P₀ is true if P₁ and P₂ and ... and Pₙ are true”

Procedural meaning:
“To satisfy goal P₀, satisfy goal P₁
then P₂ then ... then Pₙ”.
• P₀ is called the head goal of a clause.
• Conjunction of goals P₁, P₂, ..., Pₙ forms the body of the clause.
• A clause without a body is a fact:
P means “P is true” or “goal P is satisfied”
• A clause without a head
   :- P₁, P₂, ..., Pₙ or ?- P₁, P₂, ..., Pₙ
   is a query interpreted as
   “Are P₁ and P₂ and ... and Pₙ true?” or
   “Satisfy goal P₁ then P₂ then ... then Pₙ”

Lists in Prolog

A list of terms can be represented between brackets:

\[ [a, b, c, d] \]

Its head is a and its tail is [b, c, d].

The tail of [a] is [], the empty list.

Lists may contain lists:

\[ [3, [a, 8], [x], [p, q]] \] is a list of four items.

Special form to direct pattern matching:
- The term \[ X\{Y\} \] matches any list with at least one element:
  \[ X \] matches the head of the list, and \[ Y \] matches the tail.

Structures have two interpretations:
- As **predicates** (relations):
  - presidentof(marySue, iowa).
  - prime(7).
  - between(rock, X, hardPlace).

- As **structured objects** similar to records:
  - computer(name(herky), locn(‘MLH 303’),
  - make(‘IBM’), model(‘RS6000’))
  - list(3, list(5, list(8, list(13, nil)))))

Prolog Programs

A Prolog program is a sequence of statements, called **clauses**, of the form

\[ P₀ :– P₁, P₂, ..., Pₙ. \]

Each of P₀, P₁, P₂, ..., Pₙ is an atom or structure.
A **period** terminates every Prolog clause.
• The term \([X,Y,77|T]\) matches any list with at least three elements whose third element is the number 77:
  
  \(X\) matches the first element,
  
  \(Y\) matches the second element, and
  
  \(T\) matches rest of the list after the third item.

Using these pattern matching facilities, values can be specified as the intersection of constraints on terms instead of by direct assignment.

Use variable names that are suggestive:

\([\text{Head} | \text{Tail}]\) or \([H | T]\)

Recursion

Most interesting algorithms involve repeating some group of actions.

Prolog implements repetition using recursion.

Recursion is closely related to mathematical induction, requiring two cases:

**Basis:** Solve some initial or small version of the problem directly.

**Recursion:** Assuming the algorithm works on smaller or simpler versions of the problem, solve an arbitrary instance of the problem.

**Example**

\(\text{sublist}(S,L)\) succeeds if and only if the list \(S\) is a sublist of the list \(L\).

\(\text{sublist}([a,b,c], [a,b,c,d,e])\) succeeds.

\(\text{sublist}([c,d], [a,b,c,d,e])\) succeeds.

\(\text{sublist}([b,d], [a,b,c,d,e])\) fails.

For list algorithms, the basis usually deals with an empty list, certainly the smallest list. (Some algorithms for lists do not handle the empty list; so begin with a singleton list, \([H]\).)

For the recursion step, we define the algorithm for the arbitrary list, \([H|T]\), assuming that it works correctly for its tail \(T\), a smaller list.

**Sublist basis**

The empty list is a sublist of any list.

\(\text{sublist}([\ ], L).\) \hspace{1cm} % 1

**Sublist recursion**

List \([H|T]\) is a sublist of the list \([H|U]\) if list \(T\) is a sublist of list \(U\) starting at the first position.

\(\text{sublist}([H|T], [H|U]) \leftarrow \text{initialsublist}(T, U).\) \hspace{1cm} % 2

\(\text{initialsublist}([\ ], L).\) \hspace{1cm} % 3

\(\text{initialsublist}([H|T],[H|U]) \leftarrow \text{initialsublist}(T, U).\) \hspace{1cm} % 4

Or the list \(S\) is a sublist of the list \([H|T]\) if it is a sublist of \(T\).

\(\text{sublist}(S, [H|T]) \leftarrow \text{sublist}(S, T).\) \hspace{1cm} % 5

These two cases correspond to the situation where the sublist begins at the start of the list or the sublist begins later in the list, the only two possibilities.

**Sample Executions**

\(\text{sublist}([b,c,d], [a,b,c,d,e,f])\) \hspace{1cm} % 5

because \(\text{sublist}([b,c,d], [b,c,d,e,f])\) \hspace{1cm} % 2

because \(\text{initialsublist}([c,d], [c,d,e,f])\) \hspace{1cm} % 4

because \(\text{initialsublist} ([d], [d,e,f])\) \hspace{1cm} % 4

because \(\text{initialsublist} ([\ ], [e,f])\) \hspace{1cm} % 3

\(\text{sublist}([b,d], [b,c,d])\) fails \hspace{1cm} % 2

because \(\text{initialsublist}([d], [c,d])\) fails and \hspace{1cm} % 5

because \(\text{sublist}([b,d], [c,d])\) fails % 5

because \(\text{sublist}([b,d], [d])\) fails % 5

because \(\text{sublist}([b,d], [ ])\) fails
Appendix A 9

Testing Primes in Prolog

Predicate prime
- prime(P) succeeds iff P>0 is prime.
- Assume the predicate
  - sqrt(N,S) iff S = floor(sqrt(N)).

prime(N) :- N>2, N =\= 2*(N//2),
  sqrt(N,S), okay(N,S,3).

where okay(N,S,D) succeeds iff no
odd integer M with D ≤ M ≤ S, divides
into N evenly, assuming D is odd.

Predicate okay
- okay(N,S,D) :- D>S.
- okay(N,S,D) :- N =\= D*(N//D), D1 is D+2,
  okay(N,S,D1).

 Predicate sqrt
Consider this Wren program

```
program sqrt is
  var n, sqrt, odd, sum : integer;
  begin
    read n;
    sqrt := 0; odd := 1; sum := 1;
    while sum<=n do
      sqrt := sqrt+1;
      odd := odd+2;
      sum := sum+odd
    end while;
    write sqrt
  end
```

Trace
<table>
<thead>
<tr>
<th>n</th>
<th>sqrt</th>
<th>odd</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>36</td>
<td></td>
</tr>
</tbody>
</table>

Translate the while loop into Prolog as follows:
- sqrt(N,S) :- loop(N, 0, 1, 1, S).

```
loop(N, Sqrt, Odd, Sum, Ans) :-
  Sum =< N,
  Sqrt1 is Sqrt+1,
  Odd1 is Odd+2,
  Sum1 is Sum+Odd,
  loop(N, Sqrt1, Odd1, Sum1, Ans).

loop(N, Sqrt, Odd, Sum, Sqrt) :- Sum > N.
```

This last clause returns the value in the second
parameter as the answer by unifying the last
parameter with that second parameter.

Utility Predicates

get0(N)
- N is bound to the ascii code of the next
character from the current input stream
(normally the terminal keyboard).
- When the current input stream reaches its
end of file, a special value is bound to N
and the stream is closed.
  - 26, the code for control-Z or
    -1, a special end of file value.

put(N)
- The character whose ascii code is the
  value of N is printed on the current output
  stream (normally the terminal screen).

see(F)
- The file whose name is the value of F, an
  atom, becomes the current input stream.

seen
- Close the current input stream.
write(T)
The Prolog term given by T is displayed on the current output stream.

tab(N)
N spaces are printed on the output stream.

nl
Newline prints a linefeed character on the current output stream.

abort
Immediately terminate the attempt to satisfy original query and return control to top level.

name(A,L)
A is a literal atom or a number, and L is a list of the ascii codes of the characters comprising the name of A.

?- name(A,[116,104,101]).
A = the

?- name(1994,L).
L = [49, 57, 57, 52]