Lists in Prolog

Construction

[]  empty

[F|R]  cons!

[E₁, E₂, E₃]  literal
Lists in Prolog

Selection, i.e., pattern matching

length([], 0).

"don't care"

length([_|R], N) :- length(R, M), N is M+1.

math, in Prolog

pattern matching on the left-hand side!
You-try! member

member( , ).

member( , ) :- …

?- member(3, [1,2,3,4,5]).
   true

?- member(6, [1,2,3,4,5]).
   false.

?- member(E, [1,2,3,4,5]).
   E = 1 ;
   E = 2 ;
   E = 3 ;
   E = 4 ;
   E = 5.
You-try! member

member is built in to SWI-Prolog

```
member(E, [E|_]).
member(E, [_|R]) :- member(E, R).
```

?- member(3, [1,2,3,4,5]).
true;
false.

?- member(6,[1,2,3,4,5]).
false.

?- member(E,[1,2,3,4,5]).
E = 1 ;
E = 2 ;
E = 3 ;
E = 4 ;
E = 5.
The “Zebra” Puzzle

a.k.a. The “Einstein” Puzzle

<table>
<thead>
<tr>
<th>five nationalities</th>
<th>norwegian, brit, swede, dane, german</th>
</tr>
</thead>
<tbody>
<tr>
<td>five pets</td>
<td>dog, bird, zebra, cat, horse</td>
</tr>
<tr>
<td>five cigars</td>
<td>pallmall, winfield, dunhill, rothmans, marlboro</td>
</tr>
<tr>
<td>five beverages</td>
<td>tea, coffee, milk, water, beer</td>
</tr>
<tr>
<td>five house colors</td>
<td>red, green, yellow, blue, white</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fifteen clues</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) The norwegian lives in the first house.</td>
</tr>
</tbody>
</table>

Who owns the zebra?
The “Zebra” Puzzle

all the clues

(1) The Norwegian lives in the first house.
(2) The person living in the center house drinks milk.
(3) The Brit lives in a red house.
(4) The Swede keeps dogs as pets.
(5) The Dane drinks tea.
(6) The Green house is next to, and on the left of the White house.
(7) The owner of the Green house drinks coffee.
(8) The person who smokes Pall Mall rears birds.
(9) The owner of the Yellow house smokes Dunhill.
(10) The man who smokes Marlboro lives next to the one who keeps cats.
(11) The man who keeps horses lives next to the man who smokes Dunhill.
(12) The man who smokes Winfields drinks beer.
(13) The German smokes Rothmans.
(14) The red house is to the right of the blue.
(15) The Norwegian doesn't live by the red, white, or green houses.
The “Zebra” Puzzle

Representation in Prolog: all possible houses, i.e., the “state space”

houses( [ H1, H2, H3, H4, H5 ]):-
    H1 = [ N1, P1, S1, B1, C1 ],
    H2 = [ N2, P2, S2, B2, C2 ],
    H3 = [ N3, P3, S3, B3, C3 ],
    H4 = [ N4, P4, S4, B4, C4 ],
    H5 = [ N5, P5, S5, B5, C5 ],
    perm( [N1,N2,N3,N4,N5], [norwegian, brit, swede, dane, german] ),
    perm( [P1,P2,P3,P4,P5], [dog, bird, zebra, cat, horse] ),
    perm( [S1,S2,S3,S4,S5], [pallmall, winfield, dunhill, rothmans, marlboro] ),
    perm( [B1,B2,B3,B4,B5], [tea, coffee, milk, water, beer] ),
    perm( [C1,C2,C3,C4,C5], [red, green, yellow, blue, white] ).
The “Zebra” Puzzle

Representation in Prolog: the clues

einstein(Houses) :-
    Houses = [[norwegian, _, _, _, _], _, [_, _, _, milk, _], _, _],
    member([brit, _, _, _, red], Houses),
    ... 

houses( Houses ), % it's important to have this LATE (otherwise it's 120**5)
...

(1) The Norwegian lives in the first house.
(2) The person living in the center house drinks milk.
(3) The Brit lives in a red house.
...
The “Zebra” Puzzle

Representation in Prolog: solving the puzzle

solve :-
einstein( [ H1, H2, H3, H4, H5 ] ),
write( ' first house: ' ), write(H1), nl, % nl is "newline"
write( 'second house: ' ), write(H2), nl,
write( ' third house: ' ), write(H3), nl,
write( 'fourth house: ' ), write(H4), nl,
write( ' fifth house: ' ), write(H5), nl.
What's This?

d(X, X, 1) :- !.

d(C, _, 0) :- number(C).

d(X^C, X, C*X^(C-1)) :- number(C).

d(F+G, X, U+V) :- d(F,X,U), d(G,X,V).

d(F-G, X, U-V) :- d(F,X,U), d(G,X,V).

d(F*G, X, (U*G)+(V*F)) :- d(F,X,U), d(G,X,V).

d(F/G, X, ((U*G)-(V*F))/(G^2)) :- d(F,X,U), d(G,X,V).

?- d(x, x, R).
R = 1.

?- d(3*x, x, R).
R = 0*x+1*3.
Write a Simplifier!

?- d(3*x, x, R).
R = 0*x+1*3.

?- diff(3*x, x, R).
R = 3.

Start small

Stop at 9:10am

Keep track of:

the number of simplifier rules you’ve written
the most complex expression you’ve simplified
1.1 Syntax of the Predicate Calculus

4.1 Definition. The syntax of the predicate calculus ($\mathcal{PC}$) consists of symbols and formulas as follows:

Symbols

- Parentheses: $(, )$

- Sentential connectives: $\neg$, $\lor$, $\land$, $\rightarrow$, $\leftrightarrow$

- Quantifiers: $\forall$, $\exists$

- SC letters (sentential letters): $A, B, \cdots, Z$, and any of these letters with a positive Arabic numeral subscript.

Predicate symbols: An $n$-ary predicate is an uppercase letter, $A, \cdots, Z$, with the numeral $n$ as a superscript, where $n$ denotes the arity of the predicate and $0 < n$. These uppercase letters may also have numerical subscripts. Note: We will usually omit the superscript when we know the arity of a predicate.

- Individual constants: lowercase letters $a, \cdots, r$, with or without numerical subscripts.

- Individual variables: lowercase letters $s, \cdots, z$, with or without numerical subscripts.

Formulas

The set of all predicate calculus ($\mathcal{PC}$) formulas is defined recursively, beginning with the atomic formulas.

Atomic Formula:

Any single SC letter, or an $n$-ary predicate followed by exactly $n$ symbols, each of which is either an individual constant or a variable.

Formula:

Any atomic formula, or any expression (finitely long string of symbols) that is obtainable by use of the following predicate calculus construction rules (PCCR):
Prolog is syntactic sugar for the predicate calculus.

The Semantics of Predicate Logic as a Programming Language

M. H. Van Emde and R. A. Kowalski

University of Edinburgh, Edinburgh, Scotland

ABSTRACT
Sentences in first-order predicate logic can be accurately interpreted as programs. In this paper, the operational and denotational semantics of predicate logic programs are defined, and the connections with the proof theory and model theory of logic are investigated. It is concluded that operational semantics is a part of proof theory and that denotational semantics is a special case of operational semantics.

Key words and phrases: predicate logic as a programming language, semantics of programming languages, combinatory theory, programming, operational and denotational semantics, 3.1 resolution, type characterization.

1 Introduction

Predicate logic plays an important role in many formal models of computer programs [3, 14, 17]. Here we are concerned with the interpretation of predicate logic as a programming language [5, 10]. The Prolog system (for PROGRAMming in LOGic), based upon the procedural interpretation, has been used for several ambitious programming tasks.
Prolog
(proofs in)
Predicate Calculus
\[ p : \neg p \]

\[ \lambda \]

Racket
\[ \lambda \]
Calculus
\[ (\lambda (x) (x x)) (\lambda (x) (x x)) \]