UNIVERSITY OF YORK

BA Degree Examinations 1998

DEPARTMENT OF LANGUAGE AND LINGUISTIC SCIENCE

L333: Introduction to Prolog

Time allowed: $1\frac{1}{2}$ hours

Answer ALL questions

Model Answers

(1) What problems arise in expressing symmetric relations in Prolog and why? Exemplify your answer. (4 marks)

The logical definition of a symmetric predicate would look like this

$$\forall x, y[predicate(x, y) \supset predicate(y, x)]$$

The naive counterpart of such a definition in Prolog would be the following:

```
    predicate(X, Y) :-
        predicate(Y, X).
```

in conjunction with with a set of basic facts such as the following:

```
    predicate(a, b).
    predicate(c, d).
```

The problem with this definition is that, if Prolog is forced to backtrack over these clauses, it will fail to terminate and will cycle over and over again through the same set of answers. This is because the first, recursive clause will continue to call itself repeatedly.

A solution to this nontermination is to use two differently named predicates: one for the facts and one for the rule, as follows.

Continued
Turn over

```
predicate(X, Y):-
    predicate_1(Y, X).
predicate(X, Y):-

    predicate_1(X, Y).

predicate_1(a, b).
predicate_1(c, d).
```

```
With these definitions, a call to predicate/2 will terminate as soon
as all the calls to predicate\_1/2 facts have been succeeded.
```

(2) Why is the relation `brother/2` not correctly expressible in the database subset of Prolog? (3 marks)

```
Because, in normal English usage, the definition of ‘brother’ requires
that the two individuals who stand in the relation be distinct from
one another -- you cannot be your own brother.

In logical terms this requires an inequality:
```

$$\forall x, y[brother(x, y) \supset \ldots \wedge \neg x = y \wedge \ldots]$$

```
This inequality is not expressible in database (or pure) Prolog
because it requires the use of the cut.
```

(3) Assume a Prolog database containing facts for `male/1`, `female/1` and `parent/2`. In Prolog, define the following relations. (8 marks)

(a) `daughter/2`

```
% X is daughter of Y
    daughter(X, Y) :-
        female(X),
        parent(Y, X).
```

(b) `father/2`

```
% X is father of Y

    father(X, Y) :-
        male(X),
        parent(X, Y).
```

(c) `sister/2`

```
% X is sister of Y

sister(X, Y) :-
        female(X),
        parent(Z, X),
        parent(Z, Y),
        \+ X = Y.
```

(d) `grandmother/2`

```
% X is grandmother of Y

    grandmother(X, Y) :-
        female(X),
        parent(X, W),
        parent(W, Y).
```

(4) State which of the following pairs of Prolog expressions unify and (if they do) what the result of their unification is. (8 marks)

Continued
Turn over

(a)	`a`	`a`	
	`yes`	`a`	identical constants unify
(b)	`a`	`b`	
	`no`		different constants do not unify
(c)	`a`	`B`	
	`yes`	`a`	a constant and a variable unify
(d)	`A`	`B`	
	`yes`	`A (or B)`	two variables unify and become instantiated to a single value
(d)	`male(X)`	`male(phil)`	
	`yes`	`male(phil)`	
(e)	`parent(X, chas)`	`parent(liz, Y)`	
	`yes`	`parent(liz, chas)`	
(f)	`parent(X, chas)`	`parent(liz, X)`	
	`no`		X cannot be simultaneously instantiated to chas and liz

(5) (a) Explain what is meant by the Prolog notation `[X|Y]`? (2 marks)

```
This expression unifies with the head and tail of a list.  X will
unify with the first item on the list and Y will unify with the
remainder of the list (itself a list).
```

(b) What are the values returned by the following Prolog goal? (2 marks)
`?- [X|Y] = [this, is, a, list]?`

```
X = this
Y = [is, a, list]
```

(c) What are the values returned by the following Prolog goal? (2 marks)
`?- [X|Y] = [this]?`

```
X = this
Y = [ ]
```

(6) (a) Exemplifying with the predicate `ancestor/2`, explain what is meant by recursion. (4 marks)

```
A recursive predicate is one which uses itself in its own
definition.  Recursive predicates contain at least two clauses, 1)
```

Continued

```
the base, which is not recursive and 2) the recursion.

Exemplifying with ancestor/2, we have

the base:

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

and the recursion:

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

(b) Define in Prolog a predicate `member/2` which is true if its first argument is contained in the list which is its second argument. (4 marks)

```
the base:

    member(Item, [Item|_ ]).

the recursion:

    member(Item, [_|List]):-
        member(Item, List).
```

(c) Define in Prolog a predicate `append/3` all three of whose arguments are lists and which is true if its third argument is a list made up of the first argument followed by the second argument. (4 marks)

```
the base:

    append([], List, List).

the recursion:

    append([H|Tail], List, [H|Result]):-
        append(Tail, List, Result).
```

(7) Explain the following Prolog calls:

(a) (4 marks)

```
| ?- current_op(X, Y, -).
```

```
X = 500,
Y = yfx ?

yes

This goal tells us that the minus sign is defined as an infix
operator which is left-associative (yfx), and that its precedence
is 500, which means that it has a lower precedence than the
multiplication operator below and therefore that, in an expression
containing both operators, multiplication will be evaluated before
subtraction.
```

(b) (4 marks)

```
| ?- current_op(X, Y, *).

X = 400,
Y = yfx ?

yes

This goal tells us that the multiplication sign is also defined as
a left-associative infix operator, with a precedence of 400, which
means that it has a higher precedence than the subtraction
operator in (a) above.
```

(8) Given the information in question (7) above, provide answers to the following questions:

(a) What will be the result returned by the following Prolog goal? (2 marks)

```
| ?- X is 1 * 2 + 3.

X = 5.

i.e. 1*2 = 2 and 2+3 = 5
```

(b) What would the expression `1 * 2 + 3` look like if written in normal Prolog predicate/argument syntax? (2 marks)

```
    +(*(1,2),3)
```

(9) Define in Prolog a predicate `length/2` whose first argument is a list and whose second argument is an integer and which is true if its second argument is the number of items contained in its first argument. (4 marks)

```
length([], 0).

length([H|T], Length):-
    length(T, FirstLength),
    Length is FirstLength + 1.
```

(10) (a) What is the difference between the following predicate and the one you defined in answer to question (6b) above? (3 marks)

```
memberchk(I, [I|_]):- !.
memberchk(I, [_|T]):-
     memberchk(I, T).
```

```
This predicate contains a cut (!/0) as the body of the
nonrecursive clause.  This cut ensures that, once that clause
has been called and has succeeded, Prolog will be unable to
backtrack into the second (recursive) part of the definition.
```

(b) What will be the difference in Prolog's responses to calls to `member/2` and `memberchk/2` with the arguments `(X, [a,b,c])`? (3 marks)

```
The call

| ?-  member(X, [a, b, c]).

will produce the following responses:

 X = a? ;
 X = b? ;
 X = c? ;
 no

The call

| ?-  memberchk(X, [a, b, c]).

will produce the following responses:

 X = a? ;
 no
```

(11) Write a Prolog program which will produce the following result, in which the material following `|:` is supplied by the user. (15 marks)

Continued
Turn over

```
| ?-  prettyprint.
|: [this, is, a, list].
this is a list.
yes


prettyprint:-
    read(Input),
    writelist(Input).

writelist([Last]):-
    write(Last),
    write('.'),
    nl.

writelist([First|Rest]):-
    write(First),
    write(' '),
    writelist(Rest).
```