INST0072: Exercises for Lecture 7

Exercise 7-1

Given the following program [view | download]:

```
likes(dave, pete).
likes(dave, sally).
likes(eric, dave).
likes(mike, pete).

likes(john, X) :-
    \+ likes(dave, X).

likes(andy, X) :-
    \+ likes(john, X).
```

predict how Prolog answer the following queries:

```
?- likes(dave, X).
?- likes(john, X).
?- likes(andy, X).
?- likes(john, andy).
?- likes(andy, andy).
?- likes(andy, pete).
?- likes(andy, mike).
?- likes(P, Q).
```

[Answer]

Exercise 7-2

Add the following 'person/1' facts to the program in Exercise 7-1 [view | download]:

```
person(dave).
person(pete).
person(sally).
person(eric).
person(mike).
person(john).
person(andy).
```

Then modify the definition of 'likes/2' so that the program reproduces the following input/output:

```
?- likes(dave, X).
X = pete ;
X = sally.

?- likes(john, X).
X = dave ;
X = eric ;
X = mike ;
X = john ;
X = andy.

?- likes(andy, X).
X = pete ;
X = sally ;
false.

?- likes(P, Q).
P = dave,
Q = pete ;
P = dave,
Q = sally ;
P = eric,
Q = dave ;
P = mike,
Q = pete ;
P = john,
Q = dave ;
P = john,
Q = eric ;
P = john,
Q = mike ;
P = Q, Q = john ;
P = john,
Q = andy ;
P = andy,
Q = pete ;
P = andy,
Q = sally ;
false.
```

[View Example Solution | Download Example Solution]

Exercise 7-3

Add the facts

```
teaches(eric, xml).
teaches(bill, xml).
teaches(sally, algebra).
teaches(bob, java).

takes(chris, xml).
```

to the program 'happyTeacher.pl' [view | download] from Lecture 5. Then add definitions for the following predicates (adding other predicates, facts and clauses as necessary):

- The predicate 'ecstatic/1' - a teacher is ecstatic if he/she teaches hardworking, and only hardworking, students.
- The predicate 'lucky/1' - a teacher is lucky if he/she teaches a course that hardworking, and only hardworking, students are taking.
- The predicate 'confused/1' - a teacher is confused if he/she teaches a course that no students are taking.
- The predicate 'sad/1' - a teacher is sad if he/she is not teaching any hardworking students.

Your program should reproduce the following input/output:

```
?- happy(Teacher).
Teacher = frank ;
Teacher = sally ;
Teacher = eric ;
Teacher = bill ;
false.

?- ecstatic(Teacher).
Teacher = eric ;
false.

?- lucky(Teacher).
Teacher = eric ;
Teacher = bill ;
false.

?- confused(Teacher).
Teacher = sally ;
false.

?- sad(Teacher).
Teacher = bob.
```

[View Example Solution | Download Example Solution]

Exercise 7-4

If the definition of 'female/1' in the program 'mothersAndFathers.pl' [view | download] from Lecture 2 is replaced with the following

```
female(X) :-
    \+ male(X).
```

predict how Prolog answer the query

```
?- female(P).
```

Run the query with the modified program to see if your prediction was correct.

[View Modified Program | Download Modified Program]

Exercise 7-5

If the definition of 'female/1' in the program 'mothersAndFathers.pl' [view | download] from Lecture 2 is replaced with the following

```
female(X) :-
    person(X),
    \+ male(X).

person(X) :-
    father(X, Y).
person(X) :-
    father(Y, X).
person(X) :-
    mother(X, Y).
person(X) :-
    mother(Y, X).
```

predict how Prolog answer the query

```
?- female(P).
```

Run the query with the modified program to see if your prediction was correct.

[View Modified Program | Download Modified Program]

Exercise 7-6

Predict the output of the program

```
a :-
    b,
    c,
    write('1').
a :-
    d,
    write('2').

b :-
    c,
    d,
    write('3').

c :-
    write('4').
c :-
    write('5').

d :-
    write('6').
```

with the query

```
?- a.
```

[View Program | Download Program]

Exercise 7-7

Predict the output of the program

```
a :-
    b,
    c,
    write('1'),
    fail.
a :-
    d,
    write('2').

b :-
    c,
    !,
    d,
    write('3').

c :-
    write('4'),
    fail.
c :-
    write('5').

d :-
    write('6').
```

with the query

```
?- a.
```

[View Program | Download Program]

Exercise 7-8

Add an extra 'jumpOff' action to the 'monkeyAndBananaWithLoopCheck.pl' program [view | download] from Lecture 7, representing the action of the monkey jumping off the box. Test the program with various different starting positions for the monkey, the box, and choices for whether the monkey is initially on the box or not. Some example queries are:

```
?- is_plan_without_loop(state(atDoor, onFloor, atWindow, hasNot), state(_, _, _, has), Plan).
?- is_plan_without_loop(state(atWindow, onBox, atWindow, hasNot), state(atDoor, _, _, has), Plan).
```

[View Example Solution | Download Example Solution]

Exercise 7-9 [challenging!]

Use the inbuilt 'read/1' and 'write/1' predicates to write a program that asks the user to type in an 'a' character at the keyboard, and repeatedly re-prompts the user until the 'a' character is input correctly. Your program should reproduce the following input/output:

```
?- go.
Type in "a" followed by a full stop and RETURN: d.
Incorrect input!
Type in "a" followed by a full stop and RETURN: s.
Incorrect input!
Type in "a" followed by a full stop and RETURN: z.
Incorrect input!
Type in "a" followed by a full stop and RETURN: a.
Thankyou!
true ;
false.
```

[View Example Solution | Download Example Solution]

Exercise 7-10 [challenging!]

The wolf, sheep and hay problem. A man wishes to take a wolf, a sheep and some hay from the east bank to the west bank of a river. There is a boat, but it can't fit all three things in together. The man must not leave the wolf and the sheep alone on a river bank, or the wolf will eat the sheep. Similarly, he cannot leave the sheep and the hay alone on a river bank, or the sheep will eat the hay. Write a Prolog program that can plan a series of boat trips to get all three things safely across to the west bank of the river.

HINT: there are many ways to represent this problem. The example answer here uses a structure like this to represent the initial state:

```
state(hay(eastBank), man(eastBank), sheep(eastBank), wolf(eastBank))
```

and a structure like this to represent the action of taking only the sheep to the west bank:

```
take_to(hay(no), sheep(yes), wolf(no), westBank)
```

[View Example Solution | Download Example Solution]

Exercise 7-11

The wolf, sheep and hay problem with a smaller boat. A man wishes to take a wolf, a sheep and some hay from the east bank to the west bank of a river. There is a boat, but it can only take the man and one other thing at a time. The man must not leave the wolf and the sheep alone on a river bank, or the wolf will eat the sheep. Similarly, he cannot leave the sheep and the hay alone on a river bank, or the sheep will eat the hay. Write a Prolog program that can plan a series of boat trips to get all three things safely across to the west bank of the river.

HINT: This problem requires only a small change to the example answer for Exercise 4-10

[View Example Solution | Download Example Solution]

Page maintained by Rob Miller. Last updated: 3/10/2019.
