Assignment 2: A simple reflexive agent in Prolog
Group 27
30/10/2022

1.

?- parent(joost,X).
X = sacha ;
X = leon.

By executing the query **parent(joost,X).** Prolog will start looking from top to bottom of the list if there are any matching 'Facts' with the pridcate of 'parent'. The predicate parent contains two arguments (Y,X). In this query the Y=joost and the X will be matched by unification prolog. As a result we get leon and sacha matched to X. The resolution step in this clause **parent(joost,X)** the literal X will substitute the possible names. Therefore: (joost,sacha),(joost,leon) = (joost,X).

2.

?- isChild(X).
X = sacha ;
X = leon ;
X = sacha ;
X = leon ;
X = sofie ;
X = merlijn ;
X = fien ;
X = joost.

isChild (X): −parent (Y , X). This rule defines that if there is a clause with a predicate parent that has 2 arguments(Y,X), the variable named X is the child of variable Y which is the parent in our case. Prolog X substituates all constants which apply for the following condition: parent(..,X).
We get duplicates because Prolog looks through the list for the predicate ‘parent’ and gives us the second argument (X).

3.

?- brother(X,Y).
X = sacha,
Y = leon ;
X = leon,
Y = sacha ;
X = sacha,
Y = leon ;
X = leon,
Y = sacha ;
X = merlijn,
Y = sofie ;

X = joost,
Y = fien ;
false.
For each predicate 'parent' that contains ‘brothers’ we get a result of X being a brother of Y and Y being a brother of X as well. Both X and Y are male in this instance. This is how we get double entries.

?- sister(X,Y).
X = sofie,
Z = merlijn ;
false.

4.

female(fien).
We do not get a sister(X,Y) X = fien and Y = joost because fien is missing in the database. We can see in the database that Peter has a daughter Fien and a son Joost.

5.

PL file

6.

PL file

7.

?- family(sandrine,sofie).
False.
By looking at the family rule we can see that if a certain Y is a family of X it needs to atleast fullfill one of the conditions of the family rule. In this case none of the conditions are fullfilled. And we can see that this is true because sandrine and sofie are not blood related.

8.

?- family(sandrine,Y).
Y = sacha ;
Y = leon ;
False.
By executing this query Prolog checks for each condition if there is a Y which is a match to sandrine. And if there is a match it returns us the family memebers of sandrine.

9.

?- family(leon,peter).
true .

In this case the grandchild rule is applied. This one is pretty easy as we have already made the rule grandparent that helps us find the grandparents of each child. So in the grandchild rule we switch the two variables and we get the child's grandparent.

10.

?- family(sofie,sacha).
true .
The rule that gives us a result is the cousin(X,Y) rule. In that rule it searches for parents with the same parent thus the grandparent of two different children (X and Y), after that we say that X and Y can not be the same person because they have to be brothers or sisters. In this way we get the children of each parent related to each other but the program also shows us the siblings.

11.

PL file

12.

adjacent(L):-robot(X),link(X,L).
The adjacent rule gets the current position of the robot (X) and looks for a match (X) in the links (X,L) database if there is one then adjacent is true. That means that X and L are neighbours and a movement from X to L is possible.
If we choose robot(3) and execute the query **adjacent(L).** we get an output of L = 4 and L = 6 and that's true considering the links and the provided graph.

move(L):-adjacent(L),retract(robot(_)),assertz(robot(L)).
Considering the existing links in the current postition of the robot the move rule moves the robot to the next possible position. For example using robot(1) and executing move(L). we get L = 2 and this is the only possible move from current position which is point 1.

13.

suggest(L):-adjacent(L),goal(L).
If the next possible move (L) has a link to the goal then it suggests the L to reach the goal.
If we add two goals for example 5 and 7, we move the robot to position 6 and execute the query **suggest(L).** we do get both L=5 and L=7 as a possible end goal so this rule works for multiple goals.

14.

PL file

15.

In this example Prolog appends [1,2] = X and [3,4] = Y, and joins them in to the list of Z.

16.

PL file

17.

PL file

18.

It will have a infinite loop because 1 ->2->3->1. The link 3 ->1 will be created first and thus suggest(X) will suggest the first path that has a node adjecent. It will not look further down the line.