SAI assignment 2

Ryan Sleeuwaegen, Sjouk Ketwaru
s3122166, s3287297

31 oktober 2022

Question 1

Query:
parent(joost,X).

Returns:
X = sacha ; X = leon.

The explanation for this is:

- We have:
  parent(joost, sacha). parent(joost, leon).

- we ask:
  parent(joost, X)

- This unifies with:
  parent(joost, sacha), and parent(joost, leon),

  When X/sacha or X/leon.

- Subsitute sacha for X, parent(joost, X) = parent(joost, sacha).
  Subsitute leon for X, parent(joost, X) = parent(joost, leon).

Question 2

Query:
isChild(X).

Returns:
X = sacha ; X = leon ; X = sacha ; X = leon ; X = sofie ; X = merlijn ; X = fien ; X = joost.

The explanation for this is:

- We have:
  parent(joost, sacha). parent(joost, leon) . parent(sandrine, sacha).
  parent(sandrine, leon). parent(fien, sofie). parent(fien, merlijn).
  parent(peter, fien). parent(peter, joost).

- We ask:
  isChild(X):-parent(Y,X).
- This unifies with:
  parent(joost, sacha) and parent(joost, leon) and parent(sandrine, sacha) and
  parent(sandrine, leon) and parent(fien, sofie) and parent(fien, merlijn) and
  parent(peter, fien) and parent(peter, joost),

  When X/sacha and Y/joost, or X/leon and Y/joost, or X/sacha and Y/sandrine,
  or X/leon and Y/sandrine, or X/sofie and Y/fien, or X/merlijn and Y/fien,
  or X/fien and Y/peter, or X/joost and Y/peter.

- Substitute sacha for X and joost for Y,
  isChild(X):-parent(Y,X) = parent(joost, sacha).
  Substitute leon for X and joost for Y,
  isChild(X):-parent(Y,X) = parent(joost, leon).
  Substitute sacha for X and sandrine for Y,
  isChild(X):-parent(Y,X) = parent(sandrine, sacha).
  Substitute leon for X and sandrine for Y,
  isChild(X):-parent(Y,X) = parent(sandrine, leon).
  Substitute sofie for X and fien for Y,
  isChild(X):-parent(Y,X) = parent(fien, sofie).
  Substitute merlijn for X and fien for Y,
  isChild(X):-parent(Y,X) = parent(fien, merlijn).
  Substitute fien for X and peter for Y,
  isChild(X):-parent(Y,X) = parent(peter, fien).
  Substitute joost for X and peter for Y,
  isChild(X):-parent(Y,X) = parent(peter, joost).

Question 3

Added to the database:
brother(X, Y):-male(X), dif(X,Y), parent(Z,X), parent(Z,Y).
sister(X, Y):-female(X), dif(X,Y), parent(Z,X), parent(Z,Y).

Query:
brother(X,Y).

Returns:
X = sacha, Y = leon ; X = leon, Y = sacha ; X = sacha, Y = leon ; X = leon, Y = sacha ;
X = merlijn, Y = sofie ; X = joost, Y = fien.

The explanation for this is:

- We have:
  male(joost). male(sacha). male(leon). male(merlijn). male(peter).
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y).

- This unifies with:
  parent(joost, sacha), parent(joost, leon), male(sacha), dif(sacha,leon) and
  parent(joost, leon), parent(joost, sacha), male(leon), dif(leon,sacha) and
  parent(sandrine, sacha), parent(sandrine, leon), male(sacha), dif(sacha,leon) and
  parent(sandrine, leon), parent(sandrine, sacha), male(leon), dif(leon,sacha) and
  parent(fien, merlijn), parent(fien, sofie), male(merlijn), dif(merlijn,sofie) and
  parent(peter, joost), parent(peter, fien), male(joost), dif(joost,fien).

  When X/sacha, Y/leon and Z/joost,
  When X/leon, Y/sacha and Z/joost,
  When X/sacha, Y/leon and Z/sandrine,
  When X/leon, Y/sacha and Z/sandrine,
  When X/merlijn, Y/sofie and Z/fien,
  When X/joost, Y/fien and Z/peter.

- Substitute sacha for X, leon for Y and joost for Z,
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(joost,
  sacha), parent(joost, leon), male(sacha), dif(sacha,leon).

  Substitute leon for X, sacha for Y and joost for Z,
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(joost, leon), parent(joost, sacha), male(leon), dif(leon,sacha).

  Substitute sacha for X, leon for Y and sandrine for Z,
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(sandrine, sacha), parent(sandrine, leon), male(sacha), dif(sacha,leon).

  Substitute leon for X, sacha for Y and sandrine for Z,
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(sandrine, leon), parent(sandrine, sacha), male(leon), dif(leon,sacha).

  Substitute merlijn for X, sofie for Y and fien for Z,
  brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(fien, merlijn), parent(fien, sofie), male(merlijn), dif(merlijn,sofie).

Substitute joost for X, fien for Y and peter for Z,
brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), dif(X,Y) = parent(peter, joost), parent(peter, fien), male(joost), dif(joost,fien).

Query:
sister(X,Y).

Returns:
X = sofie, Y = merlijn ; X = fien, Y = joost ;

The explanation for this is:

- We have:
  female(sofie). female(sandrine). female(fien).
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  sister(X,Y):- parent(Z,X), parent(Z,Y), female(X), dif(X,Y).

- This unifies with:
  parent(fien, sofie), parent(fien, merlijn), female(sofie), dif(sofie,merlijn). parent(peter, fien), parent(peter, joost), female(fien), dif(fien,joost).

  When X/sofie, Y/merlijn and Z/fien,
  When X/fien, Y/joost and Z/peter,

- Substitute sofie for X, merlijn for Y and fien for Z,
  sister(X,Y):- parent(Z,X), parent(Z,Y), female(X), dif(X,Y) = parent(fien, sofie), parent(fien, merlijn), female(sofie), dif(sofie,merlijn).

  Substitute fien for X, joost for Y and peter for Z,
  sister(X,Y):- parent(Z,X), parent(Z,Y), female(X), dif(X,Y) = parent(peter, fien), parent(peter, joost), female(fien), dif(fien,joost).

Some entries are double because there are 2 parents registered like sandrine and joost which the code goes through 2 times. And some combinations are double because some brothers of each other so both of them are male so the code sees them as two seperate entities.

Question 4

The important fact missing from the database was:
female(fien).

Question 5

Added to the database:
cousin(X, Y):-male(X), parent(A, Y), parent(B, X), (sister(A, B) ; brother(A, B)).

Question 6

Added to the database:
sibling(X, Y):-dif(X,Y), parent(Z,X), parent(Z,Y).
ancestor(Z, X):-parent(Z, X).
ancestor(Z, X):-parent(Z, A),ancestor(A, X).
family(X, Y):-ancestor(X, Y) ; ancestor(Y,X) ; sibling(X, Y); (ancestor(Z,X), ancestor(Z, Y)), dif(X,Y).

Question 7

Query:
family(sandrine, sofie).

Returns:
false.

The explanation for this is:

- We have:
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  family(X, Y):-ancestor(X,Y) ; ancestor(Y,X) ; sibling(X, Y); (ancestor(Z,X), ancestor(Z,Y)), dif(X,Y).

  ancestor(Z, X):-parent(Z, X). ancestor(Z, X):-parent(Z, A), ancestor(A, X).

- This unifies with:
  family(sandrine, sofie):- ancestor(sandrine, sofie) ; ancestor(sofie, sandrine) ; sibling(sandrine, sofie); (ancestor(Z,sandrine), ancestor(Z, sofie)), dif(sandrine,sofie).

  ancestor(Z, sofie):-parent(Z, sofie).
  ancestor(Z, sofie):-parent(Z, A), ancestor(A, sofie)

  When X/sandrine, Y/sofie, Z/any ancestor in the tree, A/any ancestor in the tree.

- Substitute sandrine for X, sofie for Y and any ancestor for Z,
  family(X, Y):- ancestor(X, Y) ; ancestor(Y,X) ; sibling(X, Y); (ancestor(Z,X), ancestor(Z, Y)), dif(X,Y).

  ancestor(Z, X):-parent(Z, X) = family(sandrine, sofie):- ancestor(sandrine, sofie) ; ancestor(sofie, sandrine) ; sibling(sandrine, sofie); (ancestor(Z,sandrine), ancestor(Z, sofie)), dif(sandrine,sofie).
  ancestor(sandrine, sofie):-parent(sandrine, sofie). ancestor(Z, sofie):-parent(Z, A), ancestor(A, sofie)

This will continue until all ancestors and siblings are checked and eventually will give that there is no link (bloodline) between sandrine and sofie, hence the query will return false.

Question 8

Query:
family(sandrine, X).

Returns:
X = sacha, X = leon.

The explanation for this is:

- We have:
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  family(X, Y):-ancestor(X, Y) ; ancestor(Y,X) ; sibling(X, Y); (ancestor(Z,X), ancestor(Z, Y)), dif(X, Y).

  ancestor(Z, X):-parent(Z, X). ancestor(Z, X):-parent(Z, A), ancestor(A, X).

- This unifies with:
  family(sandrine, X):- ancestor(sandrine, X) ; ancestor(X, Sandrine) ; sibling(Sandrine, sofie); (ancestor(Z,sandrine), ancestor(Z, X)), dif(sandrine, X).

  ancestor(Z, sandrine):-parent(Z, sandrine).
  ancestor(Z, X):-parent(Z, A), ancestor(A, X).

  When X/sandrine, Y/X and Z/ is any ancestor in the tree, A/ is any ancestor in the tree.

Because there are no parents of sandrine in the knowledge base the ancestor will not return true, however because Z is anyone in the knowledge base it will return true for sacha and leon who are bloodrelated.

Question 9

Query:
family(leon, peter).

returns:
true.

The explanation for this is:

- We have:
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  family(X, Y):-ancestor(X, Y) ; ancestor(Y,X) ; sibling(X, Y) ; (ancestor(Z,X), ancestor(Z, Y)), dif(X,Y).

  ancestor(Z, X):-parent(Z, X). ancestor(Z, X):-parent(Z, A), ancestor(A, X).

- This unifies with:
  family(leon, peter):- ancestor(leon, peter) ; ancestor(peter, leon) ; sibling(leon, peter); (ancestor(Z,leon), ancestor(Z, peter)), dif(leon,peter).

  ancestor(Z, leon):-parent(Z, leon).
  ancestor(Z, X):-parent(Z, A), ancestor(A, X).

  When X/leon, Y/peter and Z/ is any ancestor in the tree, A/ is any ancestor in the tree.

Because peter is an ancestor of leon, the query family(leon, peter) will return true, this because it goes through the recursive ancestor until it finds the link between peter and leon.

Question 10

Query:
family(sofie,sacha).

Returns:
true.

The explanation for this is:

- We have:
  parent(joost,sacha). parent(joost,leon). parent(sandrine,sacha).
  parent(sandrine,leon). parent(fien,sofie). parent(fien,merlijn).
  parent(peter,fien). parent(peter,joost).

- We ask:
  family(X, Y):-ancestor(X, Y) ; ancestor(Y,X) ; sibling(X, Y) ; (ancestor(Z,X) , ancestor(Z, Y)), dif(X,Y).
  ancestor(Z, X):-parent(Z, X). ancestor(Z, X):-parent(Z, A), ancestor(A, X).

- This unifies with:
  family(sofie,sacha):-ancestor(sofie,sacha) ; ancestor(sacha,sofie) ; sibling(sofie,sacha) ; (ancestor(Z,sofie) , ancestor(Z,sacha)) , dif(sofie, sacha).

  ancestor(Z, sofie):-parent(Z, sofie).
  ancestor(Z, sofie):-parent(Z, A), ancestor(A, sofie).

  When X/sofie, Y/sacha, Z/ is any ancestor in the tree, A/ is any ancestor in the tree.

Sofie's ancestor is Fien, Sacha's ancestor is Joost, and since Fien and Joost are siblings the query family(sofie,sacha) returns true.

Question 11

Added to the database:
link(1,2). link(2,3). link(3,4). link(3,6). link(6,5). link(6,7).

Question 12

Added to the database:
adjacent(L):- robot(A), link(A,L).
move(L):- adjacent(L), retract(robot(_)), assertz(robot(L)).

First of all, adjacent(L) which uses the placing of the robot with Robot(A) and then checks which links are adjacent to the robot, with link(A,L), where L is the adjacent link and A is the location of the robot. Second of all, move(L), which uses the adjacent(L) to find if the link is next to the robot, retract(robot(_)) to remove the current position of the robot, assertz(robot(L)) to move the robot to the new position L.

Question 13

Added to the database:
suggest(L):- goal(L), adjacent(L).

First of all, goal(L) is the goal which you want to aim for, and adjacent checks which links are adjacent to the current position of the robot, and with this predicate suggest(L) looks at which of the links leads directly to the goal. if the goals is not within one link it will return false. Suggest(L) is also complete because it always gives an answer, it will never go in to a loop where it is never ending, say there is no link to go to then adjacent will return false and because adjacent returns false suggest(L) will also return false, hence the predicate is complete.

Question 14

Added to the database:
path:-robot(X), goal(Y), hulp(X,Y).
hulp(X,Y):-link(X,Y).
hulp(X,Y):-link(X,A), hulp(A,Y).

Question 15

Query:
append([1,2], [3,4], X)

Returns:
X = [1,2,3,4]

The explanation for this is:

- We have:
  append([ ],X,X).

- We ask:
  append([X | Y], Z, [X | W]):- append(Y, Z, W).

- This unifies with:
  append([1,2], [3,4], X)

  When Y/[1,2], Z/[3,4], W/X

- substitute [1,2] for X, [3,4] for Y,
  append([X | Y], Z, [X | W]):- append(Y, Z, W). = append([[1,2] | Y], [3,4], [[1,2,3,4] | W]):- append([1,2], [3,4], W).

Question 16

Added to the database:
path(P):-robot(X), goal(Y), link(X, A), hulp(A,Y,P).
hulp(X,X,[X]):-!.
hulp(X,Y,[X | List]):-link(X,A), hulp(A,Y,List).

Question 17

Added to the database:
suggest(L):-robot(X), goal(Y), link(X, A), hulp(A,Y,[L | P]).

Hulp here is the same as at question 16.

Question 18

Query:
suggest(X).

Returns:
ERROR: Stack limit (1.0Gb) exceeded
The explanation for this is:

- We have:
  robot(1). goal(5). link(1,2). link(2,3). link(3,1).

- We ask:
  suggest(L):-robot(X), goal(Y), link(X,A), hulp(A,Y,[L—P]).
  hulp(Z,Z,[Z]):-!.
  hulp(X,Y,[X—List]):-link(X,A), hulp(A,Y,List).

- This unifies with:
  suggest(X):-robot(1), goal(5), link(1,2), hulp(2,5,[X—P]).

  hulp(2,5,[2]):-!.
  hulp(2,5,[2—List]):-link(2,3), hulp(3,5,List).

  hulp(3,5,[3]):-!.
  hulp(3,5,[3—List]):-link(3,1), hulp(1,5,List).

  hulp(1,5,[1]):-!.
  hulp(1,5,[1—List]):-link(1,2), hulp(2,5,List).

  hulp(2,5,[2]):-!.
  hulp(2,5,[2—List]):-link(2,3), hulp(3,List).

  When X/2, Y/5
  When X/3, Y/5
  When X/1, Y/5
  When X/2, Y/5

What you see here is that prolog get's stuck in a loop where the program goes from 1 to 2, 2 to 3, and then from 3 back to 1. This is not everything that happens, since prolog first checks all other possible paths like 1 to 4, or 1 to 6.