Assignment 1 – with sample solutions

Due: Friday, 12.5.2017, 15:59 via Git

For help, contact alp-staff@lists.iai.uni-bonn.de (staff only) or
alp-course@lists.iai.uni-bonn.de (staff and participants).

Submit results into the folder "assignment01/" of the git repository of your group. For task 1 submit your implemented predicate as a file names "task1.pl" At the bottom of the file add a comment containing console output that shows all results of a successful test run.

For each other task submit your answers as either a .txt or .pdf file named according to the task number, e.g. "task2.pdf" or "task3.txt"

Task 1. *Friends* (9 Points)

Write a Prolog predicate that solves the following logic puzzle:

1. Tick, Trick and Track are friends.
2. One friend is 15, one 17, and one 18 years but we do not know who has which age.
3. One friend's last name is Chang.
4. Miss Yang is three years older than Tick.
5. The person whose last name is Thatcher is 17 years old.

Tip 1: *The condition that X is bigger by 3 than Y is written in Prolog as "X is Y+3".*

Tip 2: *Consider what the puzzle tells you about the three friends and think of a suitable term structure representing a person. Then represent each of the three friends by a person term with variables for the values that are unknown. Represent our little "world" of three friends by a list holding the three (incomplete) persons. Then use 'member(Person, List)' to search the list for a person that fulfills one of the hints given in the second to fifth sentence of the puzzle. If you do this for each hint and also consider tip 1 you have the complete predicate that solves the puzzle.*

Solution:

```
task_2_puzzle(FRIENDS):-
          FRIENDS = [ person(tick,_,_),
                      person(trick,_,_),
                      person(track,_,_)
          ],
         member(person(_,_,15), FRIENDS),
         member(person(_,_,17), FRIENDS),
         member(person(_,_,18), FRIENDS),
         member(person(_,thatcher,17), FRIENDS),
         member(person(_,chang,_), FRIENDS),
         member(person(_,yang,A), FRIENDS),
         member(person(tick,_,B), FRIENDS),
         A is B+3.
```

```
?- task_2_puzzle(FRIENDS).
Friends = [person(tick, 'Chang', 15), person(trick, 'Thatcher', 17), person(track, 'Yang', 18)] ;
Friends = [person(tick, 'Chang', 15), person(trick, 'Yang', 18), person(track, 'Thatcher', 17)] ;
false.
```

The fact that there are two solutions tells you that the problem was not specified in enough detail to give you sufficient information for a unique solution. Clearly, the family name and age of 'tick' can be determined but for 'trick' and 'track' there are two options.

If we would slightly change the task, renaming tick to Laura, trick to John and track to Anna, (or would explicitly tell you that tick and track are female and trick is male) the problem would have a unique solution because the additional information about the sex of the persons would be sufficient to disambiguate the different options:

```
task_2_puzzle_extended(FRIENDS):-
         FRIENDS = [ person(tick,_,_,female),
                     person(trick,_,_,male),
                     person(track,_,_,female)
         ],
        member(person(_,_,15,_), FRIENDS),
        member(person(_,_,17,_), FRIENDS),
        member(person(_,_,18,_), FRIENDS),
        member(person(_,thatcher,17,_), FRIENDS),
        member(person(_,chang,_,_), FRIENDS),
        member(person(_,yang,A,female), FRIENDS),
        member(person(tick,_,B,_), FRIENDS),
        A is B+3.
```

```
?- task_2_puzzle_extended (FRIENDS).
FRIENDS = [person(tick, chang, 15, female),
           person(trick, thatcher, 17, male),
           person(track, yang, 18, female)] ;
false.
```

Task 2. *Declarative semantics* (4 Points)

Assume that the information that a class or interface extends another class or interface by is represented the predicate extends/2 and that the subtype/2 predicate is defined recursively based on extends/2 as follows:

```
        extends(a, b).
        extends(c, d).
        extends(d, e).

        subtype(X,Y) :- extends(X,Y).
        subtype(X,Y) :- extends(X,Z), subtype(Z,Y).
```

For the above program write down

a) (2 Points) its translation to first order logic (quantified implications).
b) (2 Points) its model (its logical consequences).

Tip: See Chapter 2 of the lecture slides.

Solution:

a) extends(a,b) ^ extends(c,d) ^ extends(d,e) ^
∀X,Y(subtype(X,Y) <= extends(X,Y)) ^
∀X,Y .∃Z (subtype(X,Y) <= extends(X,Z) ^ subtype(Z,Y))

b) {extends(a,b), extends(c,d), extends(d,e),
subtype(a,b), subtype(c,d), subtype(d,e),
subtype(c,e)}

Task 3. *Declarative semantics* (2 Points)

```
        extends(class(a),class(b)).
        extends(class(c),class(d)).
        extends(class(d),class(e)).

        subtype(X,Y) :- extends(X,Y).
        subtype(X,Y) :- extends(X,Z), subtype(Z,Y).
```

Write down the model of the above, slightly modified, program.

Solution:

M = { extends(class(a),class(b)),
extends(class(c),class(d)),
extends(class(d),class(e)),
subtype (class(a),class(b)),
subtype (class(c),class(d)),
subtype (class(d),class(e)),
subtype (class(c),class(e)) }

Task 4. *Declarative semantics* (5 Points)

```
        natural(0).
        natural( s(X) ) :- natural(X).
```

a) (2 Points) Write down the model of the above program.
b) (1 Points) What difficulty did you encounter in step a)?
c) (2 Points) Compare this task to Task 3 and try to make a general statement about the effect of function symbols in logic programs.

Solution:

a) M = { natural(0), natural(s(0)), … }

b) The model consists of facts containing nested natural/1 terms of arbitrary depth.

c) Function symbols can make the model infinite (1 point), if they are used such that arguments in the head of a recursive clause are "bigger" than arguments in the recursive call (1 point). Then terms can grow infinitely during the fix point iteration so that there actually is no fix point (this sentence is just an additional explanation of the previous one – no extra points).

Task 5. *Unification* (3 Points)

Write down a unifier for each successful unification (for getting 0,5 extra points per unifier, provide a *most general* unifier – see slides 26 to 28 from Chapter 3). If the unification doesn't succeed explain why.

a) likes(calvin,hobbes)=likes(X,Y)
b) likes(calvin,hobbes)=likes(X,susie)
c) father(Jim, father(X))=grandfather(john, jane)
d) append([A,B,C], [D,E,F], G)=append([h,i,j], [k,l,m], [N|O])
e) [a,[b|H]|C]=[a,b,c,d]
f) [[X,Y],e|[y,z]]=[A,B,C,D]

a) likes(calvin,hobbes)=likes(X,Y)
{ X ← calvin , Y ← hobbes}

b) likes(calvin,hobbes)=likes(X,susie)
No unifiability (hobbes and susie are two different atoms)

c) father(Jim, father(X))=grandfather(john, jane)
No unifiability (the function symbols differ)

d) append([A,B,C], [D,E,F], G)=append([h,i,j], [k,l,m], [N|O])
{ A ← h , B ← i , C ← j, D ← k, E ← l, F ← m, G ← [N|O]) }

e) [a,[b|H]|C]=[a,b,c,d]
Fails because the list [b|H] cannot be unified to the constant b.

f) [[X,Y],e|[y,z]]=[A,B,C,D]
{ A ← [X,Y], B ← e, C ← y, D ← z }