Assignment 3 – Logic and Planning

Preliminaries

The programming language for this assignment is Prolog. The assignment should be submitted through ilearn2 no later than the 13th of October at 23.59 (CET).

Logic

In the file "PFAI_Assignment_3.zip" found in the ilearn2 course page, there are one files: 'logic.pl'.

Harry Potter

In 'logic.pl' the following knowledge base is found:

```
belongs_to(harry_potter, gryffindor).
belongs_to(hermione_granger, gryffindor).
belongs_to(cedric_diggory, hufflepuff).
belongs_to(draco_malfoy, slytherin).
wand(harry_potter, '11"_holly_phoenix').
wand(harry_potter, '11"_vine_dragon').
wand(harry_potter, '10"_blackthorn_unknown').
wand(harry_potter, '10"_hawthorn_unicorn').
wand(harry_potter, '15"_elder_thestral_hair').
wand(hermione_granger, '11"_vine_dragon_heartstring').
wand(hermione_granger, '13"_walnut_dragon_heartstring').
wand(cedric_diggory, '12"_ash_unicorn_hair').
wand(draco_malfoy, '10"_hawthorn_unicorn_hair').
wand(draco_malfoy, '15"_elder_thestral_hair').
patronus(harry_potter, stag).
patronus(hermione_granger, otter).
boggart(harry_potter,dementor).
boggart(hermione_granger,failure).
boggart(draco_malfoy,lord_voldemort).
loyalty(harry_potter, gryffindor).
loyalty(harry_potter, hermione_granger).
loyalty(hermione_granger, gryffindor).
loyalty(hermione_granger, harry_potter).
loyalty(cedric_diggory, hufflepuff).
loyalty(cedric_diggory, harry_potter).
influence(harry_potter, hermione_granger).
influence(hermione_granger, harry_potter).
influence(cedric_diggory, hermione_granger).
influence(cedric_diggory, harry_potter).
influence(draco_malfoy, hogwarts).
influence(hogwarts, gryffindor).
influence(hogwarts, slytherin).
influence(hogwarts, hufflepuff).
influence(hogwarts, harry_potter).
influence(hogwarts, hermione_granger).
influence(hogwarts, cedric_diggory).
influence(hogwarts, draco_malfoy).
```

Given the knowledge base write down the **all answers** to the **following questions**, in a **separate document** (note that with ";" multiple answers are generated via backtracking). There are also predicates for collecting all solutions, using setoff/3, findall/3 and bagof/3. See (for information of how to use them):
https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/mpg_002dref_002dfindall.html#mpg_002dref_002dfindall

```
?- belongs_to(X, hufflepuff).
?- loyalty(cedric_diggory, X).
?- patronus(X, Y), boggart(X, failure).
?- findall(X,(loyalty(X, Y), loyalty(Y, X)), L).
```

Given the knowledge base above **formulate the following questions in Prolog** (write down all answers in your document):

1) Does it exist a wand that has had two different owners?
2) Who influences hermoine granger?
3) Find out if there exist some, who has influence over something, that it also belongs to.
4) **Update the knowledge base** with a transitive influence rule named trans_influence/2, that denotes a transitional relationship w.r.t. influence, i.e., X rel Y <- X rel Z $\wedge$ Z rel Y. Now ask the last question again (Find out if there exist some who has influence over something that it also belongs to) write down your answer in your document.
5) Find out if there exist any entity that has (transitional) influence over themself
6) Does it exist anything that has (transitional) influence over anything else and the latter is **not loyal** to the former (and they cannot be the same object).
7) Does it exist anything that has (transitional) influence over anything else and the latter is **loyal** to the former (and they cannot be the same object).

Sets, Unification and Terms

The next part of the assignment is to **write predicates that handle sets**. Define the predicates using recursion, i.e., usage of foreach/2, fromto/4 etc. are not allowed. Note that you must write your own support predicates if you need them, e.g., if you need the functionality of member/2 write your own predicate for that, called for example m_member/2, as this is a built-in predicate which cannot be re-defined. Hence, do not use any libraries. With these **restrictions** in mind **define following predicates**:

```
%to_set(+List, -Set) Transforms a list of atoms into a set

?- to_set([a,b,b,a],S).
S = [a,b] ?

%union(+L1, +L2, -S3) S3 is the union of L1 and L2

?- union([a,b,b,a],[c,d,d,c,e,f], S).
S = [a,b,c,d,e,f] ?

%intersection(+L1, +L2, -S3) S3 is the intersection of L1 and L2

?- intersection([a,b,b,a],[c,b,b,c,e,f], S).
S = [b] ?

%diff(+L1, +L2, -S3) S3 is the difference of L1 - L2

?- diff([a,b,b,a],[c,b,b,c,e,f], S).
```

```
S = [a] ?
```

```
%subset(+L1, +L2) True if L1 is a subset of L2
```

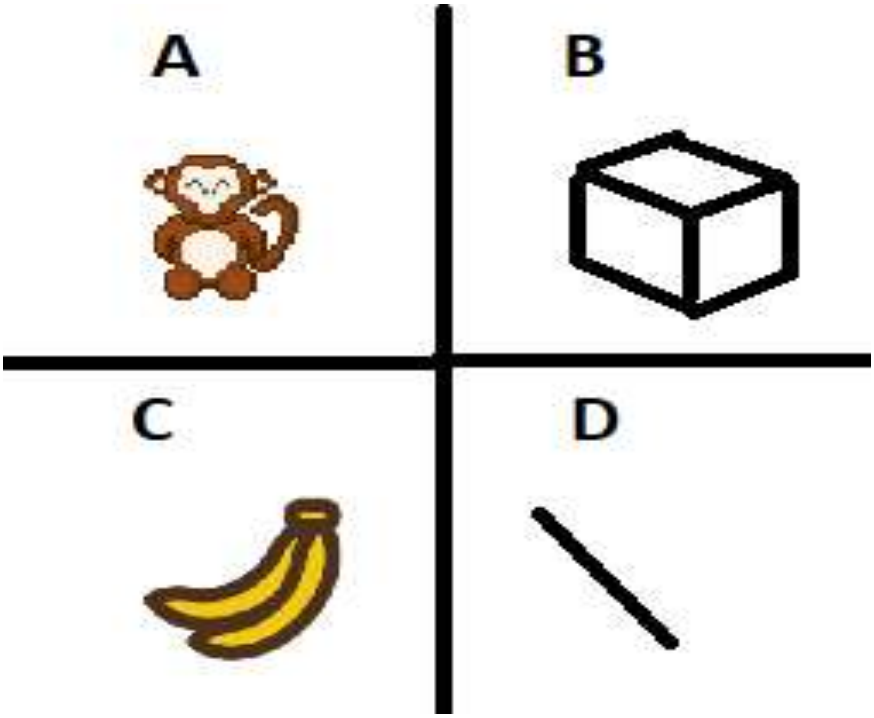
```
?- subset([b,b,f],[c,b,b,c,e,f]).
yes
```

Look at the terms that shall be **unified** (below), first do the unification in your head, then use Prolog to do the unification for you. If you experience a mismatch, reflect on it.

1) [X,Y,Z|L] and [a|Zs]
2) [a,b] and [X,Y,Z]
3) f(g(X),a,Y) and f(g(b),Y,Z)
4) p(q(X,Y),r([a,b])) and p(q(a,b),r(X,Y))
5) p(s(X),a,Z,Z) and p(Y,X,r(Y),r(s(a)))

Monkey and banana problem

A monkey is in a room, divided into 4 sections (a, b, c and d). Suspended from the ceiling is a bunch of bananas, beyond the monkey's reach. However, in the room there are also a box and a stick. The ceiling is just the right height so that a monkey standing on a box could knock the bananas down with the stick. The monkey knows how to move around, push and carry things around and wave a stick in the air. What is the best sequence of actions for the monkey to get gold of the bananas? See figure below for an initial setting of the problem.



Last in the file logic.pl, is a definition of the above problem. But the code is **missing** some **vital parts** and **definitions**. In the code there are comments about what needs to be complemented. Your job is to **complete the code** so that the problem can be solved. Here it is possible that some **help predicates** that you wrote for handling sets can be re-used (please test them thoroughly first so that you know that they work properly).

Summary of assignment

Your groups assignment should contain the code that you have produced:
'logic.pl' – containing additional code according to the instructions above.
text document – Notes from your experiments, in the same order as they are requested in this document.
Add both files into a zip file with the last names of all group members, like
'LastName1_LastName2_ass3.zip'