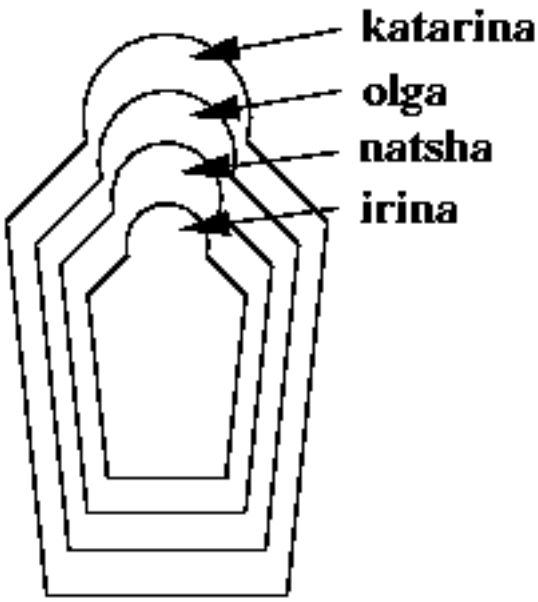
Übungsaufgaben

Logik in der Praxis - Logikprogrammierung (Prolog)

Abgabe bis 21.10.08

Exercise 3.1 (1 Punkt)
Do you know these wooden Russian dolls, where smaller ones are contained in bigger ones? Here is schematic picture of such dolls.



First, write a knowledge base using the predicate `directlyIn/2` which encodes which doll is directly contained in which other doll. Then, define a (recursive) predicate `in/2`, that tells us which doll is (directly or indirectly) contained in which other doll. E.g. the query `in(katarina,natasha)` should evaluate to true, while `in(olga, katarina)` should fail.

Exercise 3.2 (1 Punkt)
Define a predicate `greater_than/2` that takes two numerals in the notation that we introduced in this lecture (i.e. 0, succ(0), succ(succ(0)) ...) as arguments and decides whether the first one is greater than the second one. E.g:

```
?- greater_than(succ(succ(succ(0))),succ(0)).
yes
?- greater_than(succ(succ(0)),succ(succ(succ(0)))).
no
```

Exercise 3.3 (3 Punkte)
Binary trees are trees where all internal nodes have exactly two childres. The smalles binary trees consist of only one leaf node. We will represent leaf nodes as `leaf(Label)`. For instance, `leaf(3)` and `leaf(7)` are leaf nodes, and therefore small binary trees. Given two binary trees `B1` and `B2` we can combine them into one binary tree using the predicate tree: `tree(B1,B2)`. So, from the leaves `leaf(1)` and `leaf(2)` we can build the binary tree

`tree(leaf(1), leaf(2))`. And from the binary trees `tree(leaf(1), leaf(2))` and `leaf(4)` we can build the binary tree `tree(tree(leaf(1), leaf(2)), leaf(4))`.
Now, define a predicate `swap/2`, which produces a mirror image of the binary tree that is its first argument. For example:

```
?- swap(tree(tree(leaf(1), leaf(2)), leaf(4)),T).
T = tree(leaf(4), tree(leaf(2), leaf(1))).
yes
```

Exercise 3.4 (2 Punkte)
In the lecture, we saw the predicate

```
descend(X,Y) :- child(X,Y).
descend(X,Y) :- child(X,Z),
                descend(Z,Y).
```

Could we have formulated this predicate as follows?

```
descend(X,Y) :- child(X,Y).
descend(X,Y) :- descend(X,Z),
                descend(Z,Y).
```

Compare the declarative and the procedural meaning of this predicate definition.

Hint: What happens when you ask the query `descend(rose,martha)`?

Exercise 3.5 (3 Punkte)
We have the following knowledge base:

```
directTrain(forbach,saarbruecken).
directTrain(freyming,forbach).
directTrain(fahlquemont,stAvold).
directTrain(stAvold,forbach).
directTrain(saarbruecken,dudweiler).
directTrain(metz,fahlquemont).
directTrain(nancy,metz).
```

That is, this knowledge base holds facts about towns it is possible to travel between by taking a **direct** train. But of course, we can travel further by “chaining together” direct train journeys. Write a recursive predicate `travelBetween/2` that tells us when we can travel by train between two towns. For example, when given the query

```
travelBetween(nancy,saarbruecken).
```

it should reply `yes'.
It is, furthermore, plausible to assume that whenever it is possible to take a direct train from A to B, it is also possible to take a direct train from B to A. Can you encode this in Prolog? You program should e.g. answer `yes' to the following query:

```
travelBetween(saarbruecken,nancy).
```

Do you see any problems you program may run into?