02156 Exercises-04

Jørgen Villadsen

2021

Exercise 1

Consider the following formula: $(p \to q \to r) \to (p \to q) \to p \to r$

Use refutation and the systematic construction of a semantic tableau.

State whether this shows that the formula is valid or not.

Exercise 2

Write a program `tautology(+XFml)` that succeeds if and only if `XFml` is a tautology (`XFml` is a formula in external format, for example `p => p`, which is furthermore a simple tautology).

Use a substantial modification of the program `truthtable` available in the file `logic.pl` on CampusNet (Program files folder in the top folder).

```
truthtable(XFml) :-
  to_internal(XFml,Fml),
  get_atoms(Fml,Atoms),
  write_tt_title(Fml,Atoms),
  generate(Atoms,V),
  tt(Fml,V,TV),
  write_tt_line(Fml,V,TV),
  fail.
truthtable(_).
```

Use the program `tautology` to check whether $p \to q \to p$ and $(p \to q) \to p$ are tautologies.

Exercise 3

Write a deterministic program `delete(?List1,?Elem,?List2)` that succeeds if and only if `List1` with all occurences of `Elem` deleted unifies with `List2`.

Make sure that the program is always deterministic, that is, it does not succeed more than once.

Exercise 4

Consider the use of lists as sets (no duplicate elements in the lists).

Write deterministic programs `intersection(+Set1,+Set2,?Set3)` and `union(+Set1,+Set2,?Set3)` that unifies `Set3` with the intersection and union, respectively, of `Set1` and `Set2`.

In case the basic predicate `member` is used, would it be appropriate to use the deterministic predicate `membercheck` instead?

```
?- intersection([a,b,c],[b,d],Z).

Z = [b] ;

No

?- union([a,b,c],[b,d],Z).

Z = [a, c, b, d] ;

No
```

Exercise 5

Provide a proof in the Gentzen system $\mathcal{G}$ and in the Hilbert system $\mathcal{H}$ of the formula $((p \to q) \to p) \to p$.

Hint: Consider Theorem 3.20 and Theorem 3.31 (it is not simple).

Exercise 6

First load the file `map.pl` with the program `test` from CampusNet (Program files folder in the top folder).

```
test(X) :- map(X), colouring(X).

map([
  (A,[I,S,G],austria),
  (B,[F,H,G],belgium),
  (D,[G],denmark),
  (F,[E,I,S,B,G],france),
  (G,[F,A,S,H,B,D],germany),
  (H,[B,G],holland),
  (I,[F,A,S],italy),
  (P,[E],portugal),
  (E,[F,P],spain),
  (S,[F,I,A,G],switzerland)
]).

% Suggestion: Use numbers 0,1,2,3 as colours
```

Write a predicate `colouring(+Map)` that succeeds if and only if `Map` can be coloured such that no two adjacent countries get the same colour.

The argument `Map` is an incomplete data structure: Logical variables are present and can be instantiated.

```
?- test(X).

X = [ (0, [1, 2, 1], austria),
      (0, [3, 2, 1], belgium),
      (0, [1], denmark),
      (3, [0, 1, 2, 0, 1], france),
      (1, [3, 0, 2, 2, 0, 0], germany),
      (2, [0, 1], holland),
      (1, [3, 0, 2], italy),
      (1, [0], portugal),
      (0, [3, 1], spain),
      (2, [3, 1, 0, 1], switzerland) ]

Yes
```

Use a predicate `members(?List1,?List2)` that succeeds if and only if all elements in `List1` are members of `List2`.

```
?- members([c,b],[a,b,c,d]).

Yes

?- members([c,e,b],[a,b,c,d]).

No
```

Start by writing the predicate `members` (it is quite simple).