**Logical reasoning and programming**, lab session III  
(October 16, 2017)

**III.1** Study the unification algorithm, see the slides, and prove that it always terminates.

**III.2** Find a most general unifier for \( f(X, f(X, Y)) \) and \( f(g(Y), f(g(a), Z)) \).

**III.3** Try Prolog using an online version of Flach’s book [Simply Logical](#). You can go through examples in [Part I, Chapter 1](#) and play with them directly using embedded [SWISH](#). In particular, query the knowledge base using various defined predicates and change their arguments from input to output and vice versa.

**III.4** What do you get if you try to list all underground lines using a query `?-connected(_,_,L).`? Why?

(It is possible to obtain a set of solutions using `sefof/3` that will be discussed later on, see `?-setof(Z,X^Y^connected(X,Y,Z),L).`)

**III.5** Is it possible to produce a symmetric variant of `connected/3` from Flach’s book by adding rule

\[
\text{connected}(X,Y,L) :- \text{connected}(Y,X,L).
\]

to your program? What is the problem with that rule and how would you fix it? (Try to define a new predicate `connected_sym/3`.)

**III.6** Write a predicate `member/2` such that `member(X,L)` holds iff `X` is a member of a list `L`, e.g., `member(b,[a,b,c])`.

**III.7** Write a predicate `b2a/2` whose arguments are lists. It holds that `b2a(L1,L2)` iff `L1` contains the same number of `b`s as `L2` contains `a`s, e.g., `b2a([b,b],[a,a])`.

**III.8** Write a predicate `append/3` whose arguments are lists. It holds `append(L1,L2,L3)` iff `L3` is a concatenation of lists `L1` and `L2`, e.g., `append([a],[b,c],[a,b,c])`.  

1