

# Foundations of Logic Programming

## Tutorial 4 (on December 1st)

Lukas Schweizer

WS 2017/18

### Exercise 4.1:

Show with the help of the Prolog tree how the cut is used in the following program,

```
r(a).
r(b).
q(a) ← r(X), !, p(a).
q(f(X)) ← r(X).
p(X) ← r(X).
p(f(X)) ← q(X), !, r(X).
p(g(X)) ← r(X).
```

and where the query `?- p(X).` is taken. What would happen without the cut?

### Exercise 4.2:

Consider the following program together with the query `?- r(X).`

```
q(b).
r(a).
s(b).
p(X) ← q(X), s(X), !.
p(X) ← r(X).
r(X) ← s(X).
r(X) ← p(X), !, q(X).
```

- Show with the help of the Prolog tree how the *cut* is used, i.e. indicate explicitly, if branches are eliminated from the tree.
- Give the output in the order of the computation.

**Exercise 4.3:**

The built-in predicate `fail/0`, fails when Prolog encounters it as a goal. Thus, it can be viewed as an instruction for backtracking. On the other hand, the cut predicate `!`, blocks backtracking.

Define the predicate `neg/1` which allows you to express *negation as failure*.