Assignment 9
Introduction to Computational Logic, SS 2011

Prof. Dr. Gert Smolka, Dr. Chad Brown
www.ps.uni-saarland.de/courses/cl-ss11/

Read in the lecture notes: Chapters 4-5

Note: It is very important to do all the examples in the lecture notes and the exercises below in the system Coq.

Exercise 9.1 Prove the following lemma.

```
Lemma nat_dis (x : nat) :  S x <> O.
```

First do the proof analogous to the proof of *bool_dis* using auxiliary functions and the *change* tactic. Then do the proofs again using Coq's tactic *discriminate e*.

Exercise 9.2 The member constructors of an inductive type are always injective provided the inductive type is not a proposition. Lemma *S_injective* proves this fact for *nat*. Prove an analogous result for the member constructor of *pair*.

```
Lemma pair_injective (X Y : Type) (x x' : X) (y y' : Y) :
pair x y = pair x' y'  -> x = x' /\ y = y'.
```

First do the the proof analogous to the proof of *S_injective* using the lemma *f_equal*. Then do the proofs again using Coq's tactic *injection e*.

Exercise 9.3 Prove $bool \neq nat$. Hint: Use as discriminating predicate a predicate saying that given three members of a type at least two of them must be equal.

Exercise 9.4 Prove the following variant of Kaminski's equation.

```
Lemma Kaminski2 (f g : bool -> bool) (x : bool) :
f (f (f (g x))) = f (g (g (g x))).
```

Exercise 9.5 Give three inductive proofs of $\forall x{:}\,nat.\ Sx \neq x$:

a) With the tactic *induction*.

b) With the function *nat_Ep*.

c) With the tactic *fix*.

Exercise 9.6 Prove $\forall n.\ even\ n = negb\,(even\,(S\ n))$ with the tactic *induction*. You will need a lemma. The proof is difficult since the recursion of *even* takes off two applications of the constructor S while *induction* takes off only one application of S.

Exercise 9.7 Write a function that computes factorials with primitive recursion. Prove the correctness of your functions.

Exercise 9.8 Recall the definitions $AF : nat \to Prop$ and $K : nat \to \forall n : nat, AFn$ from the lecture notes. Give an alternative definition $K' : nat \to \forall n : nat, AFn$ using *nat_E* such that the following lemmas are provable using *reflexivity*.

```
Lemma K_K'_5 (c : nat) : K c 5 = K' c 5.
reflexivity. Qed.

Lemma K_K'_7 (c : nat) : K c 7 = K' c 7.
reflexivity. Qed.
```

Exercise 9.9 (Projections) Define a function $P : \forall n : nat.\ nat \to AF\ n$ satisfying the following defining equations.

$$\begin{aligned} P\ O\ k &= k \\ P\ (S\,n)\ O\ x &= K\ x\ n \\ P\ (S\,n)\ (S\,k)\ x &= P\ n\ k \end{aligned}$$

Prove that your function satisfies the defining equations. Also check that the term $P\,4\,2$ reduces to $fun\ \_\ \_\ x\ \_ : nat \Rightarrow x$.

Exercise 9.10 Prove that $nat \to nat$ is uncountable. First do a direct proof in the style of *uncountable_nat_bool*. Then prove the claim with *Cantor_generalized*.

Exercise 9.11 Prove that *option nat* is countable.

Exercise 9.12 Prove the following lemmas.

```
Lemma le_O {x} : x <= O -> x = O.
Lemma le_S x : x <= S x.
Lemma le_irr x : ~ x < x.
```

Exercise 9.13 Prove the following variants of *le_trans*.

```
Lemma le_lt_trans {x} y {z} : x <= y -> y < z -> x < z.
Lemma lt_le_trans {x} y {z} : x < y -> y <= z -> x < z.
```