Assignment 3
Introduction to Computational Logic, SS 2012

Prof. Dr. Gert Smolka, Dr. Chad Brown
www.ps.uni-saarland.de/courses/cl-ss12/

Read in the lecture notes: Chapter 3

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

Exercise 3.1 In the lecture notes we describe the elimination rule for disjunction in detail and relate it to a Coq tactic. Make sure that you can discuss each introduction and elimination rule in this fashion.

Exercise 3.2 Fill in the underlines in the following proof script.

```
Goal forall (X : Type) (p q : X -> Prop),
(exists x, p x /\ q x) -> exists x, p x.

Proof. intros X p q A.
apply (ExE _ A). intros x B.
apply (AndE B). intros C _.
apply (ExI _ x).   exact C. Qed.
```

Exercise 3.3 Formulate the introduction and elimination rules for disjunctions as lemmas and use the lemmas to prove the commutativity of disjunction.

Exercise 3.4 The tactics *reflexivity* and *rewrite* implement the following proof rules for equations.

$$\frac{}{s = s} \qquad\qquad \frac{s = t \qquad u_t^x}{u_s^x}$$

a) Formulate a lemma *EqI* expressing the introduction rule for equations.

b) Formulate a lemma *EqE* expressing the elimination rule for equations.

c) Prove symmetry and transitivity of equality using the lemmas *EqI* and *EqE*. Do not use the tactics *reflexivity* and *transitivity*.

Exercise 3.5 Prove that the following propositions are equivalent. There are short proofs if you use *tauto*.

```
Definition XM : Prop := forall X : Prop, X \/ ~X.          (* excluded middle *)
Definition DN : Prop := forall X : Prop, ~~X -> X.         (* double negation *)
Definition CP : Prop := forall X Y : Prop, (~Y -> ~X) -> X -> Y.   (* contraposition *)
Definition Peirce : Prop := forall X Y : Prop, ((X -> Y) -> X) -> X.   (* Peirce's Law *)
```

Exercise 3.6 Prove the following goals. They state consequences of the De Morgan laws for conjunction and universal quantification whose proofs require the use of excluded middle.

```
Goal forall X Y : Prop,
XM -> ~(X /\ Y) -> ~X \/ ~Y.
Goal forall (X : Type) (p : X -> Prop),
XM -> ~(forall x, p x) -> exists x, ~p x.
```

Exercise 3.7 Prove the Drinker Paradox:

```
Lemma Drinker (X : Type) (d : X -> Prop) :
(exists x:X, x = x) -> XM -> exists x:X, d x -> forall y:X, d y.
```

Exercise 3.8 Two propositions are given below. In each case either the proposition is provable or the negation of the proposition is provable. Determine which and then do the proof in Coq. (You may use *tauto*.)

a) `forall P:Prop, exists G:Prop -> Prop, forall X Y:Prop, (X /\ P -> Y) <-> (X -> G Y)`

b) `forall P:Prop, exists F:Prop -> Prop, forall X Y:Prop, (X -> Y /\ P) <-> (F X -> Y)`

Exercise 3.9 Give an inductive definition of negation corresponding to the following introduction rule. Afterwards, prove a lemma corresponding to the following elimination rule.

$$\frac{\forall X : Prop, s \to X}{\neg s} \qquad\qquad \frac{\neg s \qquad s}{u}$$

Exercise 3.10 Consider a logical operation Q of type

```
forall X:Type, (X -> Prop) -> (X -> Prop) -> Prop
```

We will write $Qx : s.(p, q)$ as notation for

```
Q s (fun x:s => p) (fun x:s => q)
```

Consider the following introduction and elimination rules for Q.

$$\frac{v : s \qquad t_v^x \qquad u_v^x}{Qx : s.(t, u)} \qquad\qquad \frac{Qx : s.(t, u) \qquad x : s\,,\, t\,,\, u \Rightarrow v}{v}$$

a) Give a plain definition of Q using the logical connectives we have studied so far. Prove lemmas corresponding to the introduction and elimination rule above.

b) Give an inductive definition of Q with constructor corresponding to the introduction rule. Prove a lemma corresponding to the elimination rule.