Assignment 10
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 10

This assignment is accompanied by a Coq development. Study the Coq development. For the exercises below use the definitions of the Coq development. Also use rewriting with list equivalences as set up by the development.

Exercise 10.1 Prove the following goal with the tactics *intros*, *rewrite*, and *reflexivity*.

```
Goal forall A A' B B',  (A <-> A') -> (B <-> B') -> (A -> B <-> A' -> B').
```

Exercise 10.2 Coq's library *List* defines an existential version *existsb* of *forallb*. Prove the following reflection lemma.

```
Lemma existsb_ref X f A :
existsb f A <-> exists x : X, In x A /\ f x.
```

Exercise 10.3 Prove the following de Morgan lemma for *forallb* and *existsb*. Use the lemma *negb_andb* from *Bool*.

```
Definition negbfun (X : Type) (f : X -> bool) (x : X) : bool := negb (f x).

Lemma negb_forallb X (f : X -> bool) A :
negb (forallb f A) = existsb (negbfun f) A.
```

Exercise 10.4 Prove the following list equivalences.

```
Lemma swap_cons X (x y : X) A :
equi (x :: y :: A) (y :: x :: A).

Lemma shift_cons X (x : X) A B :
equi (x :: A ++ B) (A ++ x :: B).
```

Exercise 10.5 Prove that a formula is valid if and only if its negation is unsatisfiable.

```
Lemma valid_unsat s :
valid s <-> ~sat [Not s].
```

Exercise 10.6 Prove that the functions *eva* and *sat* are compatible with list inclusion.

```
Lemma eva_incl G G' a :
incl G G' -> eva a G' -> eva a G.

Lemma sat_incl G G':
incl G G' -> sat G' -> sat G.
```

Exercise 10.7 Prove the following facts about implications.

```
Lemma eva_imp_pos a s t G :
eva a (Imp s t :: G) = eva a (Not s :: G) || eva a (t :: G).

Lemma eva_imp_neg a s t G :
eva a (Not (Imp s t) :: G) = eva a (s :: Not t :: G).

Lemma sat_imp_pos s t G :
sat (Imp s t :: G) <-> sat (Not s :: G) \/ sat (t :: G).

Lemma sat_imp_neg s t G :
sat (Not (Imp s t) :: G) <-> sat (s :: Not t :: G).
```

Exercise 10.8 Understand and prove the following facts. Use the lemma *nd_sound*.

```
Lemma refut_sound G :
refut G -> ~sat G.

Lemma nd_consistency : ~ nd nil Fal.

Goal (forall G, refut G <-> ~sat G) -> forall s, valid s <-> refut [Not s].
```

Exercise 10.9 For each of the following formulas s give a tableau derivation of the clause $[\neg s]$ both with a diagram and with a Coq script.

a) $x \to y \to x$

b) $(x \to y \to z) \to (x \to y) \to x \to z$

c) $(x \to \neg y \to \bot) \to \neg\neg(x \to y)$

d) $\neg x \to \neg\neg y \to \neg(x \to y)$

Exercise 10.10 Consider formulas that also feature conjunctions and disjunctions. Extend the tableau system with rules for conjunctions and disjunctions such that the two key properties for tableau rules are satisfied and conjunctions and disjunctions are decomposed (i.e., do not appear in the premises of the rules). You should have rules for positive and negative conjunctions and for positive and negative disjunctions.

Exercise 10.11 Assume the lemmas *tab_nd* and *refut_sound* and prove the following lemma.

```
Lemma tab_sound C :
tab C -> ~sat C.
```