Semantics, WS 2011-2012: Solution for Assignment 7

Prof. Dr. Gert Smolka, Dr. Chad Brown

Read the new material in Chapter 4 of the lecture notes.

Exercise 7.1 Give the induction principles for the following inductive predicates defined in this chapter and check your results with Coq.

a) *And*

b) *Eq*

c) *EQ*

Solution to Exercise 7.1

```
Inductive And (X Y : Prop) : Prop :=
| AndI : X -> Y -> And X Y.

And_ind
     : forall X Y P : Prop, (X -> Y -> P) -> And X Y -> P

Inductive Eq (X : Type) : X -> X -> Prop :=
| EqI : forall x, Eq X x x.

Eq_ind
     : forall (X : Type) (P : X -> X -> Prop),
       (forall x : X, P x x) -> forall y z : X, Eq X y z -> P y z

Inductive EQ (X : Type) (x : X) : X -> Prop :=
| EQI : EQ X x x.

EQ_ind
     : forall (X : Type) (x : X) (P : X -> Prop),
       P x -> forall y : X, EQ X x y -> P y
```

Exercise 7.2 Give the induction principle for the following inductive predicate.

```
Inductive le2 : nat -> nat -> Prop :=
| le2x : forall x, le2 x x
| le2S : forall x y, le2 x y -> le2 x (S y).
```

Solution to Exercise 7.2

```
Inductive le2 : nat -> nat -> Prop :=
| le2x : forall x, le2 x x
| le2S : forall x y, le2 x y -> le2 x (S y).

le2_ind
     : forall P : nat -> nat -> Prop,
       (forall x : nat, P x x) ->
       (forall x y : nat, le2 x y -> P x y -> P x (S y)) ->
       forall x y : nat, le2 x y -> P x y
```

Exercise 7.3 Prove the following goal. Do not use a lemma.

```
Lemma starTR : forall x y z, star x y -> R y z -> star x z.
```

Solution to Exercise 7.3

```
Lemma starTR : forall x y z, star x y -> R y z -> star x z.

Proof. intros x y z A B. induction A ; eauto using star. Qed.
```

Exercise 7.4 Give the induction principle for the inductive predicate *star*.

Solution to Exercise 7.4

```
star_ind
     : forall P : X -> X -> Prop,
       (forall x : X, P x x) ->
       (forall x y z : X, R x y -> star y z -> P y z -> P x z) ->
       forall x y : X, star x y -> P x y
```

Exercise 7.5 We can define a reflexive transitive closure predicate *star1* with a single proper argument.

```
Inductive star1 (x : X) : X -> Prop :=
| star1R : star1 x x
| star1T : forall y z, star1 x y -> R y z -> star1 x z.
```

a) Give the induction principle for *star1*.

b) Prove that *star1* is reflexive and transitive.

c) Prove $\forall x y.\ star\, x\, y \leftrightarrow star1\, x\, y$

Solution to Exercise 7.5

```
star1_ind
    : forall (x : X) (P : X -> Prop),
      P x ->
      (forall y z : X, star1 x y -> P y -> R y z -> P z) ->
      forall y : X, star1 x y -> P y

Goal forall x, star1 x x.

Proof. eauto using star1. Qed.

Goal forall x y z, star1 x y -> star1 y z -> star1 x z.

Proof. intros x y z A B. induction B ; eauto using star1. Qed.

Lemma star1TL : forall x y z, R x y -> star1 y z -> star1 x z.
Proof. intros x y z A B. induction B ; eauto using star1. Qed.

Goal forall x y, star x y <-> star1 x y.

Proof. split.
intros A. induction A ; eauto using star1, star1TL.
intros A. induction A ; eauto using star, starTR.
Qed.
```

Exercise 7.6 Prove that taking the reflexive transitive closure preserves invariants.

```
Definition invariant {X : Type} (p : X -> Prop) (R : X -> X -> Prop) : Prop :=
 forall x y, R x y -> p x -> p y.

Goal forall (X : Type) (R : X -> X -> Prop) (p : X -> Prop),
invariant p R -> invariant p (star R).
```

Solution to Exercise 7.6

```
Goal forall (X : Type) (R : X -> X -> Prop) (p : X -> Prop),
invariant p R -> invariant p (star R).

Proof. intros X r p A x y C D. induction C ; firstorder. Qed.
```

Exercise 7.7 You may have seen $R^* := \bigcup_{n\in\mathbb{N}} R^n$ as a definition of the reflexive transitive closure. Using the function *iter*, we can express this definition in Coq.

```
Definition comp {X : Type} (R S : X -> X -> Prop) (x z : X) : Prop :=
exists y, R x y /\ S y z.
```

```
Definition stari {X : Type} (R : X -> X -> Prop) (x y : X) :=
exists n, iter n (comp R) (fun x y => x=y) x y.
```

Prove the equivalence of the inductive and the iterative definition.

```
Goal forall (X : Type) (R : X -> X-> Prop) (x y : X),
star R x y <-> stari R x y.
```

Solution to Exercise 7.7

```
Goal forall (X : Type) (R : X -> X-> Prop) (x y : X),
star R x y <-> stari R x y.

Proof. split.
intros A. induction A. exists 0. reflexivity.
   destruct IHA as [n B]. exists (S n). simpl. exists y. tauto.
intros [n A]. revert x y A. induction n ;simpl.
   intros x y A. rewrite A. now constructor.
   intros x y [x' [A B]]. apply (starT X R x x') ; now auto. Qed.
```

Exercise 7.8 Give three proofs for the following goal.

```
Goal forall r r' : rel, rap r r' -> functional r' -> functional r.
```

a) Use *firstorder*.
b) Use *eauto*.
c) Use *eapply* and *eassumption*.

Solution to Exercise 7.8

```
Goal forall r r' : rel, rap r r' -> functional r' -> functional r.

Proof. firstorder. Qed.

Goal forall r r' : rel, rap r r' -> functional r' -> functional r.

Proof. intros r r' A B s s' s'' C D. eauto. Qed.

Goal forall r r' : rel, rap r r' -> functional r' -> functional r.

Proof. intros r r' A B s s' s'' C D. eapply B; eapply A; eassumption. Qed.
```

Exercise 7.9 Download the Coq code from the lecture of November 30. Consider the correctness theorem for the compiler from commands to regular expressions.

```
Theorem correctness c :
req (sem c) (den (compile c)).
```

The proof has two directions. The proof of the first direction is given.

a) Rewrite the proof of the first direction so that it does not use *firstorder* or *eauto*.

b) Write the proof of the second direction.

Solution to Exercise 7.9

```
Theorem correctness c :
req (sem c) (den (compile c)).

Proof. split.
(* -> *)
intros s s' A. induction A.
(* Act *) simpl. reflexivity.
(* Seq *) simpl. exists s'. split; assumption.
(* IfTrue *) simpl. left. exists s. split. split.
              assumption. reflexivity. assumption.
(* IfFalse *) simpl. right. exists s. split. split.
        apply negteq. assumption. reflexivity. assumption.
(* WhileFalse *) simpl. exists s. split. now apply starR.
        split. apply negteq. assumption. reflexivity.
(* WhileTrue *)  simpl. exists s''.
      destruct IHA2 as [s3 [H3 [H4 H5]]]. subst.
      split.
      apply starS with (s' := s').
      exists s. split. split. assumption. reflexivity. assumption.
      assumption.
      split. assumption. reflexivity.
 (* <- *)
induction c; simpl; intros s s' A.
(* Act *) rewrite <- A. now constructor.
(* Seq *) destruct A as [s2 [H1 H2]]. econstructor. eapply IHc1. eassumption.
        apply IHc2. now apply H2.
(* If  *) destruct A as [[s2 [[H1 H2] H3]]|[s2 [[H1 H2] H3]]].
        subst. econstructor. assumption. apply IHc1. assumption.
        subst. apply semIfFalse. apply negteq. assumption. apply IHc2. assumption.
```

```
(* While *) destruct A as [s2 [H [H1 H2]]]. subst.
        induction H.
        apply semWhileFalse. apply negteq. assumption.
        destruct H as [s3 [[H2 H3] H4]]. subst.
        apply semWhileTrue with (s' := s'). assumption. apply IHc. assumption.
        apply IHstar. assumption.
Qed.
```

Exercise 7.10 Prove the following goal by applying the induction principle *even_ind*. Do not use the induction tactic.

```
Goal forall n, even n -> even (S n) -> False.
```

Solution to Exercise 7.10

```
Goal forall n, even n -> even (S n) -> False.
apply (even_ind (fun n => even (S n) -> False)).
intros A. now inversion A.
intros n A IHA B. inversion B. auto. Qed.
```