Semantics, WS 2011-2012:
Solution for Assignment 4

Prof. Dr. Gert Smolka, Dr. Chad Brown

Exercise 4.1 Define a family of inductive propositions that behave like disjunctions and prove that your disjunctions are equivalent to Coq's predefined disjunctions.

Solution to Exercise 4.1

```
Inductive Or (X Y:Prop) : Prop :=
| OrIL : X -> Or X Y
| OrIR : Y -> Or X Y.

Goal forall X Y:Prop, Or X Y <-> X \/ Y.
split.
intros A. destruct A. left. assumption. right. assumption.
intros A. destruct A. apply OrIL. assumption. apply OrIR. assumption.
Qed.
```

Exercise 4.2 Consider the following inductive proposition.

```
Inductive decp : Prop -> Prop :=
| decp0 : forall X:Prop, ~X -> decp X
| decp1 : forall X:Prop, X -> decp X.
```

Prove the following.

```
Goal forall X:Prop, decp X <-> X \/ ~X.
```

Solution to Exercise 4.2

```
Goal forall X:Prop, decp X <-> X \/ ~X.
intros X. split.
intros H. destruct H.
right. exact H.
left. exact H.
intros H. destruct H.
apply decp1. exact H.
apply decp0. exact H.
Qed.
```

Exercise 4.3 Define a family of inductive propositions that behave like existential quantifications. Arrange your definition such that you obtain a constructor *Ex* for which you can prove

```
Goal forall (X : Type) (p : X -> Prop), Ex X p <-> exists x, p x.
```

Solution to Exercise 4.3

```
Inductive Ex (X : Type) (p : X -> Prop) : Prop :=
| ExI : forall x, p x -> Ex X p.

Goal forall (X : Type) (p : X -> Prop),
Ex X p <-> exists x, p x.

Proof. split.
intros [x A]. exists x. exact A.
intros [x A]. exact (ExI X p x A). Qed.
```

Exercise 4.4 Prove the following goal.

```
Goal forall (X : Type) (x y : X) (p : X -> Prop),
Eq X x y -> p x -> p y.
```

Solution to Exercise 4.4

```
Goal forall (X : Type) (x y : X) (p : X -> Prop),
Eq X x y -> p x -> p y.

Proof. intros X x y p A. destruct A. intros H. exact H.
Qed.
```

Exercise 4.5 Prove the following goals.

```
Goal forall n, even n -> even (pred (pred n)).

Goal forall m n, even m -> even n -> even (m+n).

Goal forall m n, even (m+n) -> even m -> even n.
```

Solution to Exercise 4.5

```
Goal forall n,
even n -> even (pred (pred n)).

Proof. intros n [|n' A]. now constructor. exact A. Qed.
```

```
Goal forall m n,
even m -> even n -> even (m+n).

Proof. intros m n A. induction A ; simpl.
now auto.
intros B. constructor. now auto. Qed.

Goal forall m n,
even (m+n) -> even m -> even n.

Proof. intros m n A B. induction B.
exact A.
inversion A. now auto. Qed.
```

Exercise 4.6 One can define evenness with a boolean function.

```
Fixpoint evenb (n : nat) : bool :=
match n with
| 0 => true
| 1 => false
| S (S n') => evenb n'
end.
```

Prove that the boolean and the inductive definition agree. The proof goes through if you generalize the claim as follows.

```
Goal forall n,
(evenb n = true <-> even n) /\
(evenb (S n) = true <-> even (S n)).
```

Solution to Exercise 4.6

```
Proof. induction n ; split.
 split ; intros A. now constructor. reflexivity.
 split ; intros A. discriminate A. now inversion A.
tauto.
destruct IHn as [[A B] _]. simpl. split ; intros C.
   constructor. now auto.
   inversion C. now auto. Qed.
```

Exercise 4.7 Prove that *leq* is transitive.

```
Goal forall x y z, leq x y -> leq y z -> leq x z.
```

Solution to Exercise 4.7

```
Proof. intros x y z A. revert z. induction A ; intros z H.
now constructor.
destruct z ; inversion H. constructor. now auto. Qed.
```

Exercise 4.8 Prove that *leq* agrees with a boolean definition of the natural order.

```
Fixpoint leqb (x y : nat) : bool :=
match x,y with
| 0, _ => true
| S _, 0 => false
| S x', S y' => leqb x' y'
end.
```

```
Goal forall x y, leqb x y = true <-> leq x y.
```

Solution to Exercise 4.8

```
Proof. split.
revert y. induction x ; intros y A. now constructor.
   destruct y. discriminate A. constructor. now auto.
intros A. induction A ; now auto. Qed.
```

Exercise 4.9 Define a recursive function

```
eq_nat : nat -> nat -> bool
```

and prove

```
Goal forall x y, eq_nat x y = true <-> x = y.
```

Solution to Exercise 4.9

```
Fixpoint eq_nat (n m:nat) : bool :=
match n,m with
| O,O => true
| S n',S m' => eq_nat n' m'
| _,_ => false
end.
```

```
Goal forall x y, eq_nat x y = true <-> x = y.
intros x. induction x.
intros [|y].
simpl. split; intros _; now reflexivity.
```

```
simpl. split; intros H; now discriminate.
intros [| y].
simpl. split; intros H; now discriminate.
destruct (IHx y) as [H1 H2].
simpl. split; intros H.
rewrite (H1 H). reflexivity.
apply H2. inversion H. reflexivity.
Qed.
```

Exercise 4.10 Define a induction predicate

```
odd : nat -> Prop
```

and prove

```
Goal forall x, odd x <-> even (S x).
```

Give the inference rules for odd. You may need to formulate and prove a lemma to use.

Solution to Exercise 4.10

```
Inductive odd : nat -> Prop :=
| odd1 : odd 1
| oddS : forall n, odd n -> odd (S (S n)).

Lemma lem410 : forall x, even x -> odd (S x).
intros x A. induction A.
now constructor.
constructor. assumption.
Qed.

Goal forall x, odd x <-> even (S x).
split.
intros A. induction A.
constructor. now constructor.
constructor. assumption.
destruct x as [|x].
intros H. now inversion H.
intros H. inversion H as [|y H1 H2]. apply lem410. assumption.
Qed.
```

Exercise 4.11 Define an inductive predicate

```
rel : exp -> nat -> Prop
```

and prove

```
Goal forall e n, rel e n <-> evalExp e = n.
```

Give the inference rules for *rel* (write $e \Downarrow n$ for *rel e n*).

Solution to Exercise 4.11

```
Inductive rel : exp -> nat -> Prop :=
| relC : forall n, rel (Const n) n
| relB : forall b e1 e2 n1 n2,
           rel e1 n1 -> rel e2 n2 ->
           rel (Binop b e1 e2) (evalBinop b n1 n2).

Goal forall e n, rel e n <-> evalExp e = n.

Proof. split ; intros A.
induction A ; simpl ; congruence. (* induction on e more tedious *)
rewrite <- A. clear n A. (* essential, otherwise induction generalizes A *)
induction e ; simpl ; constructor ; assumption. Qed.
```

The inference rules:

$$\frac{}{n \Downarrow n} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{\mathit{Binop\ Plus}\, e_1\, e_2 \Downarrow n_1 \cdot n_2} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{\mathit{Binop\ Times}\, e_1\, e_2 \Downarrow n_1 + n_2}$$