软件理论基础与实践

IndProp: Inductively Defined Propositions

胡振江 熊英飞
北京大学

复习：ev定义

```
Inductive ev : nat -> Prop :=
| ev_0 : ev 0
| ev_SS (n : nat) (H : ev n) : ev (S (S n)).
```

等价于如下逻辑推导式

$$ev\ 0$$

$$\frac{ev\ n}{ev\ (S\ (S\ n))}$$

分解证据

```
Theorem evSS_ev : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  (* E : ev (S (S n)) *)
  destruct E as [| n' E'] eqn:EE.
  - (* E : ev 0
        EE : E = ev_0 *)
Abort.
```

用destruct分解带index的定义时，index会被constructor覆盖，导致定理无法证明。

remember

```
Theorem evSS_ev_remember : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  (* E: ev (S (S n)) *)
  remember (S (S n)) as k eqn:Hk.
  (* Hk : k = S (S n)
     E: ev k *)
  destruct E as [|n' E'] eqn:EE.
  - (* Hk: 0 = S (S n)
       E: ev 0
       EE: E = ev_0 *)
    discriminate Hk.
  - (* E = ev_S n' E' *)
    injection Hk as Heq. rewrite <- Heq. apply E'.
Qed.
```

remember策略把所有S (S n)替换成变量k，之后destruct策略会自动替换该变量。

能否不用remember达到同样的效果?

```
Theorem evSS_ev_remember : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  assert (exists k, k = (S (S n))). {
    exists (S (S n)). reflexivity.
  }
  destruct H as [k Hk].
  rewrite <- Hk in E.
  (* n, x: nat
     E: ev k
     Hk: k = S (S n) *)
```

inversion

```
Theorem evSS_ev' : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  inversion E as [| n' E' Heq].
  (*  E = ev_SS n' E'*)
  apply E'.
Qed.
```

inversion策略针对归纳定义的命题进行了优化

- 对于每个constructor生成一个goal
- 将constructor的参数添加为假设（以上两项同destruct）
- 对比constructor产生的index和被分解的假设，添加等式，并智能改写目标
- 如果等式矛盾，删除对应目标

inversion

```
Theorem inversion_ex1 : forall (n m o : nat),
  [n; m] = [o; o] ->
  [n] = [m].
Proof.
  intros n m o H.
采用inversion:
  inversion H. reflexivity. Qed.
采用injection:
  injection H. intros. rewrite H0. rewrite H1.
  reflexivity. Qed.
```

Inversion可以起injection的作用（destruct会分解=）

```
Theorem inversion_ex2 : forall (n : nat),
  S n = O ->
  2 + 2 = 5.
Proof.
  intros n contra. inversion contra. Qed.
```

inversion也能替代discriminate

induction用于归纳定义的命题

```
Inductive ev : nat -> Prop :=
| ev_0 : ev 0
| ev_SS (n : nat) (H : ev n) : ev (S (S n)).
```

```
Lemma ev_even : forall n,
  ev n -> even n.
Proof.
  intros n E.
  induction E as [|n' E' IH].
  - (* E = ev_0 *) exists 0. reflexivity.
  - (* E': ev n’                    （constructor参数）
       IH: even n’                  （归纳假设）
       goal: even (S (S n'))        （重写目标）        *)
    unfold even in IH.
    destruct IH as [k Hk].
    rewrite Hk. exists (S k). simpl. reflexivity.
Qed.
```

命题对应的结构归纳法定理

```
Check ev_ind :
  forall P : nat -> Prop,
    P 0 ->
    (forall n : nat, ev n -> P n -> P (S (S n))) ->
    forall n : nat, ev n -> P n.
```

P作用在ev的参数上，ev n作为前提出现

归纳定义的关系

```
Inductive le : nat -> nat -> Prop :=
  | le_n (n : nat)                  : le n n
  | le_S (n m : nat) (H : le n m) : le n (S m).

Notation "n <= m" := (le n m).
```

等价于如下逻辑推导式

$$n \le n$$

$$\frac{n \le m}{n \le S\, m}$$

定理可以用相同方法证明

```
Theorem test_le1 :
  3 <= 3.
Proof.
  apply le_n.  Qed.

Theorem test_le2 :
  3 <= 6.
Proof.
  apply le_S. apply le_S. apply le_S. apply le_n.  Qed.

Theorem test_le3 :
  (2 <= 1) -> 2 + 2 = 5.
Proof.
  intros H. inversion H. inversion H2.  Qed.
```

关系对应的结构归纳法定理

```
Check le_ind : forall P : nat -> nat -> Prop,
  (forall n : nat, P n n) ->
  (forall n m : nat, le n m -> P n m -> P n (S m)) ->
   forall n n0 : nat, le n n0 -> P n n0.
```

P作用在le的所有参数上

关系对应的结构归纳法定理

```
Print le.
(* ====> Inductive le (n : nat) : nat -> Prop :=
             le_n : n <= n
           | le_S : forall m : nat, n <= m -> n <= S m *)
```

```
Check le_ind:
  forall (n : nat) (P : nat -> Prop),
    P n ->
    (forall m : nat, n <= m -> P m -> P (S m)) ->
    forall n0 : nat, n <= n0 -> P n0.
```

P只作用在index上

回顾eq_ind定理

```
Inductive eq (A : Type) (x : A) : A -> Prop
  :=  eq_refl : x = x
```

```
Check eq_ind :
  forall (A: Type) (x:A) (P : A -> Prop),
    P x -> forall y : A, x=y -> P y.
```

eq_ind也是生成的归纳法定理

作业

- 完成IndProp中standard非optional截止到case study（不含）之前的习题
  - 请使用最新英文版教材