



软件理论基础与实践

IndProp: Inductively Defined Propositions

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

$$\frac{ev\ 0}{ev\ (S\ (S\ n))}$$



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自动将所有S (S n)替换成0，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。



remember

```
Theorem evSS_ev_remember : forall n,  
  ev (S (S n)) -> ev n.
```

Proof.

```
intros n E.  
(* n : nat  
  E: ev (S (S n)) *)  
remember (S (S n)) as k eqn:Hk.  
(* n, k : nat  
  Hk : k = S (S n)  
  E: ev k *)
```

Show Proof.

```
(* (fun (n : nat) (E : ev (S (S n)))) =>  
  let k := S (S n) in  
  let Hk : k = S (S n)  
  := eq_refl      in ?Goal) *)
```

Remember采用let引入新的假设

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 (H:exists k, k = (S (S n))). {  
    exists (S (S n)). reflexivity.  
  }  
  destruct H as [k Hk].  
  rewrite <- Hk in E.  
  (* n, k: 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，
并自动引入和改写
(destruct会分解=)

```
Theorem inversion_ex2 : forall (n : nat),  
  S n = 0 ->  
  2 + 2 = 5.
```

Proof.

```
  intros n contra. inversion contra. Qed.
```

inversion也能替代
discriminate



复习： Induction

```
Theorem add_0_r : forall n:nat, n + 0 = n.
Proof. intros n.
(*   n : nat
 *   =====
 *   n + 0 = n
 *)
  induction n as [| n' IHn'].
  - reflexivity.
  -
    (*   n' : nat
     *   IHn' : n' + 0 = n'
     *   =====
     *   S n' + 0 = S n'
     *)
```

Induction将n的所有出现替换成对应的constructor调用
如果constructor包含递归参数，添加参数和对应归纳假设到假设区



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.



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

$$\frac{n \leq m}{n \leq S m}$$



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.
```



作业

- 完成IndProp中standard非optional截止到case study（不含）之前的7道习题
 - 请使用最新英文版教材