软件理论基础与实践

Induction: Proof by Induction

熊英飞
北京大学

多文件程序开发

- `From LF Require Export Basics.`
  - 从LF目录读取编译后的Basics.vo文件并导入
- LF目录
  - _CoqProject文件指明当前目录为LF
    - `-Q . LF`
- 编译得到Basics.vo文件
  - Proof General和CoqIDE会自动编译
    - VSCode不行
  - Linux下编译
    - Make Basics.vo或make
  - Windows下编译
    - coqc –Q . LF Basics.v

复习

- 是否能通过simpl和reflexivity证明n+0=n

```
Theorem add_0_r_firsttry : forall n:nat,
  n + 0 = n.
Proof. intros n.
(*   n : nat
 *   =========
 *   n + 0 = n
 *)
  simpl.(** [Coq Proof View]
 *   n : nat
 *   =========
 *   n + 0 = n
 *)
 (* Does nothing! *)
Abort.
```

Destruct也不行

```
Theorem add_0_r_secondtry : forall n:nat,
  n + 0 = n.
Proof.
  intros n. destruct n as [| n'] eqn:E.
  - (* n = 0 *)
    reflexivity. (* so far so good... *)
  - (* n = S n’ *)
    (* S n' + 0 = S n’ *)
    simpl.
    (* S (n' + 0) = S n’ *)
Abort.
```

结构归纳法
Structural Induction

- 数学归纳法：
  - 首先证明性质P对0成立
  - 然后证明P(k)->P(k+1)
- 结构归纳法：
  - 对于任意递归定义的类型
  - 证明P对该类型的每一个构造函数都成立
    - 如果构造函数接受该类型的参数，则假设P对参数成立
- 数学归纳法是结构归纳法在自然数上的特例

采用结构归纳法证明

```
Theorem add_0_r : forall n:nat, n + 0 = n.
(** [Coq Proof View]
 * 1 subgoal
 *
 *   ========================
 *   forall n : nat, n + 0 = n
 *)
Proof. intros n.
(** [Coq Proof View]
 * 1 subgoal
 *
 *   n : nat
 *   =========
 *   n + 0 = n
 *)
```

采用结构归纳法证明

```
induction n as [| n' IHn'].
(** [Coq Proof View]
 * 2 subgoals
 *
 *   ==============
 *   0 + 0 = 0
 *
 * subgoal 2 is:
 *  S n' + 0 = S n'
 *)
```

为变量命名，可省略

采用结构归纳法证明

```
  - reflexivity.
  - (** [Coq Proof View]
 * 1 subgoal
 *
 *   n' : nat
 *   IHn' : n' + 0 = n'
 *   ==================
 *   S n' + 0 = S n'
 *)
    simpl. rewrite -> IHn'. reflexivity.
Qed.
```

归纳假设

Induction同时作用于一个变量的多个实例

```
Theorem minus_n_n : forall n,
  minus n n = 0.
Proof.
  (* WORKED IN CLASS *)
  intros n. induction n as [| n' IHn'].
  - (* n = 0 *)
    simpl. reflexivity.
  - (* n = S n' *)
    simpl. rewrite -> IHn'. reflexivity.  Qed.
```

证明子定理

```
Theorem mult_0_plus' : forall n m : nat,
  (0 + n) * m = n * m.
Proof.
  intros n m.
  assert (H: 0 + n = n). { reflexivity. }
  rewrite -> H.
  reflexivity.  Qed.
```

证明子定理

```
Theorem mult_0_plus' : forall n m : nat,
  (0 + n) * m = n * m.
Proof. intros n m.
(** [Coq Proof View]
 * 1 subgoal
 *
 *   n, m : nat
 *   ===================
 *   (0 + n) * m = n * m
 *)
```

证明子定理

```
 assert (H: 0 + n = n).
(** [Coq Proof View]
 * 2 subgoals
 *
 *   n, m : nat
 *   ==================
 *   0 + n = n
 *
 * subgoal 2 is:
 *  (0 + n) * m = n * m
 *)
```

证明子定理

```
{ reflexivity. }
 (** [Coq Proof View]
* 1 subgoal
*
*   n, m : nat
*   H : 0 + n = n
*   ===================
*   (0 + n) * m = n * m
*)
rewrite -> H.
reflexivity.  Qed.
```

采用assert帮助rewrite定位

```
Theorem plus_rearrange_firsttry : forall n m p q : nat,
  (n + m) + (p + q) = (m + n) + (p + q).
Proof. intros n m p q.
(** [Coq Proof View]
 * 1 subgoal
 *
 *   n, m, p, q : nat
 *   ==================================
 *   n + m + (p + q) = m + n + (p + q)
 *)
```

采用assert帮助rewrite定位

```
  rewrite add_comm.
(** [Coq Proof View]
 * 1 subgoal
 *
 *   n, m, p, q : nat
 *   =================================
 *   p + q + (n + m) = m + n + (p + q)
 *)
```

- Rewrite只重写了最外层的加法

采用assert帮助rewrite定位

```
Theorem plus_rearrange : forall n m p q : nat,
  (n + m) + (p + q) = (m + n) + (p + q).
Proof.
  intros n m p q.
  assert (H: n + m = m + n).
  { rewrite add_comm. reflexivity. }
  rewrite H. reflexivity.  Qed.
```

形式化证明 vs 非形式化证明

- Coq写的是形式化证明
- 数学课上/论文里写的是非形式化证明
- 非形式化证明的作用
  - 交流——非形式化证明可以写得更简洁，更抽象，帮助交流
  - 理解——采用Coq自动证明策略很容易试出来证明，但更重要的是确保自己能理解证明

作业

- 完成Induction.v中standard非optional的4道习题
  - 请使用最新英文版教材