「計算と論理」 Software Foundations その2

五十嵐 淳
cal15@fos.kuis.kyoto-u.ac.jp

京都大学

October 13, 2015

宿題提出の手順

1. アカウント作成ページへのアクセス
   - (教科書をダウンロードした時のパスワードを入力)
   - 必要事項を入力する
2. 宿題提出システムへアクセス (要アカウント情報)
3. 提出ページへのリンクを辿り，指示に従って提出

いずれのページも講義 WWW ページからのリンク有

別ファイルの定義・定理の読み込み

```
Require Export Basics.
```

- Basics.v を「コンパイル」した Basics.vo が必要.
- 参考情報: Induction.v 冒頭
- テキストのディレクトリで make してもよい (はず)

Induction.v

- 数学的帰納法による証明 (induction タクティック)
- 証明中の証明 (assert タクティック)

Induction.v

- 数学的帰納法による証明 (induction タクティック)
- 証明中の証明 (assert タクティック)

帰納法による証明

定理: 0 は足し算の右単位元

```
Theorem plus_O_r : forall n:nat,
  n + 0 = n.
```

詰まる証明

```
Proof.
  intros n. reflexivity. (* エラー! *)
```

何が起こっているのか？

```
Proof.
  intros n. simpl. (* 左辺は計算できない *)
```

「こういう時は場合分けでしょ？」

またもや詰まる証明

```
Proof.
  intros n. destruct n as [| n'].
  - (* n = 0 *)
    reflexivity. (* so far so good... *)
  - (* n = S n' *)
```

- 場合分けをいくら続けてもキリがない!
- n より 1 小さい n' について plus_0_r が成り立っていれば…

「こういう時は場合分けでしょ？」

またもや詰まる証明

```
Proof.
  intros n. destruct n as [| n'].
  - (* n = 0 *)
    reflexivity. (* so far so good... *)
  - (* n = S n' *)
    simpl. (* また同じようなゴールが… orz *)
```

- 場合分けをいくら続けてもキリがない!
- n より 1 小さい n' について plus_0_r が成り立っていれば…

「こういう時は場合分けでしょ？」

またもや詰まる証明

```
Proof.
  intros n. destruct n as [| n'].
  - (* n = 0 *)
    reflexivity. (* so far so good... *)
  - (* n = S n' *)
    simpl. (* また同じようなゴールが… orz *)
```

- 場合分けをいくら続けてもキリがない!
- n より 1 小さい n' について plus_0_r が成り立っていれば… ⇒ 数学的帰納法

数学的帰納法

$P(n)$ を自然数 n の性質について述べた命題とする

数学的帰納法の原理

「任意の自然数 n について $P(n)$」は以下と同値

- $P(0)$ かつ
- 任意の自然数 n' について，$P(n')$ ならば $P(S\ n')$

単なる場合分けと違って，$P(S\ n')$ を示すのに，ひとつ小さい数では P が成立していること (つまり $P(n')$) を仮定してよい

- $P(n')$ を「帰納法の仮定」(induction hypothesis, IH) と呼ぶ

数学的帰納法の妥当性

個々の具体的な数 (例えば 4) について P が成立することが,

- $P(0)$ かつ
- 任意の自然数 n' について, $P(n')$ ならば $P(S\ n')$

を組み合わせて導き出せる

数学的帰納法を使った証明

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

基本的な使い方は `destruct` と同じ

- intro パターン
- `IHn'` が帰納法の仮定 (Coq が勝手に名前をつける)

日本語で書くなら…

定理: 任意の自然数 n について $n + 0 = n$.

n についての数学的帰納法による.

- $n = 0$ の場合，示すべきは $0 + 0 = 0$ だが，これは $+$ の定義により自明.
- $n = S(n')$ の場合，示すべきは $S(n') + 0 = S(n')$ であるが,

$$\begin{aligned} \text{左辺} &= S(n' + 0) && + \text{ の定義による} \\ &= S(n') && \text{帰納法の仮定による} \\ &= \text{右辺} \end{aligned}$$

なので証明終.

数学的帰納法を使った証明(2)

```
Theorem minus_diag : forall n,
  minus n n = 0.
```

今日のメニュー

Induction.v

- 数学的帰納法による証明 (induction タクティック)
- 証明中の証明 (assert タクティック)

証明中の証明

- 以前に証明した定理は他の定理の証明中で使える
- 証明中でも「サブ定理」(補題) を宣言・証明できる

$\Rightarrow$ `assert` タクティック

assert を使った(人工的な)例

```
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.
```

assert を使った(人工的な)例

```
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.
```

- `assert (0 + n = n) as H` と書いてもよい

assert の挙動

- 新たなサブゴールとして assert された命題が追加される
- 前のゴールの文脈には assert された命題が仮定として追加されている

assert の応用

そこじゃない!

```
Theorem plus_rearrange_firsttry :
  forall n m p q : nat,
    (n + m) + (p + q) = (m + n) + (p + q).
Proof.
  intros n m p q.
    (* n と m を入れ替えればいいんでしょ？ *)
  rewrite -> plus_comm.
    (* ゴールが…思ってたのと違う！ *)
```

assert の応用

```
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).
    (* この文脈での n と m の交換に特化 *)
     rewrite -> plus_comm. reflexivity.
  rewrite -> H. reflexivity. Qed.
```

こうしなきゃいけないのはどうかと思うが…

- 一応，他の手段はありますが講義ではカバーしません

宿題： 10/27 午前10:30 締切

- `Induction.v` の `basic_induction` (2), `double_plus` (2), `beq_nat_refl` (2)
- その他は随意課題
- 講義・演習に関する質問，わかりにくいと感じたこと，その他気になること，を自由に．(「特になし」はダメです．)
- 解答を書き込んだ `Basics.v`, `Induction.v` を含む zip ファイルをオンライン提出システムを通じて提出
- 友達に教えてもらったら、その人の名前を明記