「計算と論理」
Software Foundations
その7

五十嵐 淳
igarashi@kuis.kyoto-u.ac.jp

京都大学

December 3, 2012

はじめに

- これまでの章の内容: 型付関数プログラムに関する
  - 等しさ
  - 含意 (ならば)
  - 全称量化 (任意の〜について)

  を使った主張 (命題) とその正しさの証拠 (証明)
- 本章の内容:
  - 「等しさ」のような基本的・原始的な命題の表現
  - 基本命題の証拠・証明とは何か

今日のメニュー

Prop.v

- **帰納的に定義される命題**
- **証明オブジェクト**
  - **証明スクリプトと証明オブジェクト**
  - **含意 (「ならば」) と関数**
  - **証明オブジェクトに関する帰納法**
  - **偶数性**
  - **計算による定義と帰納的定義**
  - **証拠についての場合わけ・inversion**
- ...

帰納的に定義される命題

定義: 「美しい」自然数

自然数 n が美しいとは，$n = 0, 3, 5$ のいずれかであるか，他の美しい自然数の和で表されることをいう．

推論規則による「美しさ」の定義:

$$\frac{}{0 \text{ is beautiful}} \quad (\textsc{B0})$$

$$\frac{}{3 \text{ is beautiful}} \quad (\textsc{B3})$$

$$\frac{}{5 \text{ is beautiful}} \quad (\textsc{B5})$$

$$\frac{\mathbf{n} \text{ is beautiful} \qquad \mathbf{m} \text{ is beautiful}}{\mathbf{n} + \mathbf{m} \text{ is beautiful}} \quad (\textsc{BSUM})$$

「8 は美しい」ことの導出

その 1:

$$\dfrac{\dfrac{}{\text{3 is beautiful}}\,\mathrm{B3}\quad\dfrac{}{\text{5 is beautiful}}\,\mathrm{B5}}{\text{8 is beautiful}}\,\mathrm{BSUM}$$

その 2:

$$\dfrac{\dfrac{}{\text{5 is beautiful}}\,\mathrm{B5}\quad\dfrac{\dfrac{}{\text{0 is beautiful}}\,\mathrm{B0}\quad\dfrac{}{\text{3 is beautiful}}\,\mathrm{B3}}{\text{3 is beautiful}}\,\mathrm{BSUM}}{\text{8 is beautiful}}\,\mathrm{BSUM}$$

Coq での「美しさ」の定義

```
Inductive beautiful : nat -> Prop :=
  b_0   : beautiful 0
| b_3   : beautiful 3
| b_5   : beautiful 5
| b_sum : forall n m,
 beautiful n -> beautiful m -> beautiful (n+m).
```

- **一行目**: `beautiful` **は (自然数を添字とする) 命題**
  - `beautiful 0`, `beautiful 1`, ... **は (**`0 = 1` **のような) 命題**
- **残り**: **「正しい」**`beautiful` **n を定義**
  - **推論規則を記号化**

Coq での「美しさ」の証明

```
Theorem three_is_beautiful: beautiful 3.
Proof.
    apply b_3. (* 規則 B3 による *)
Qed.

Theorem eight_is_beautiful: beautiful 8.
Proof.
    apply b_sum with (n:=3) (m:=5).
      (* 規則 BSUM 中の n, m を具体化 *)
    apply b_3.
    apply b_5.
Qed.
```

今日のメニュー

Prop.v

- 帰納的に定義される命題
- 証明オブジェクト
  - 証明スクリプトと証明オブジェクト
  - 含意 (「ならば」) と関数
  - 証明オブジェクトに関する帰納法
  - 偶数性
  - 計算による定義と帰納的定義
  - 証拠についての場合わけ・inversion
- ...

型定義 vs 命題の定義

帰納的な型定義:

```
Inductive nat :  Type := ...
```

- **型** `nat` **の要素を規定**
  - `O : nat`
  - `S(O) : nat`
  - `S(S(O)) : nat`
  - ⋮

帰納的な命題定義:

```
Inductive beautiful :  nat -> Prop := ...
```

- **命題** `beautiful` **n が成立する条件を規定**
- **型定義と同様な解釈 (「要素を規定」) はできる?**

Yes! — カリー・ハワードの対応

“a : b” のふたつの読み方:

- 式 a は型 b を持つ (has type)
- a は命題 b の証明である (is a proof of)
  - 命題定義の b0 : beautiful0

カリー・ハワードの対応 (その 1)

論理の世界	計算 (プログラム) の世界
命題	型
証拠・証明	データ・式

```
b_sum : forall n m,
 beautiful n -> beautiful m -> beautiful (n+m)
```

をプログラムの世界で再解釈

$\Longrightarrow$ `b_sum` は

- **ふたつの自然数 n, m**
- `beautiful` **n 型の値，**`beautiful` **m 型の値**

の 4 引数を取る関数で，実際，

```
Check (b_sum 3 5 b_3 b_5).
b_sum 3 5 b_3 b_5
      : beautiful (3 + 5)
```

8 が美しいことの別証:

```
Theorem eight_is_beautiful': beautiful 8.
Proof.
    apply (b_sum 3 5 b_3 b_5).
Qed.
```

- 命題 `beautiful 8` と `beautiful (3+5)` は「等しい」
- `apply` タクティックの新しい使い方に注意
  - 引数は証明オブジェクトを表す式でもよい
  - (仮定も証明オブジェクトを表す式)

4引数!?

関数の型って -> で書くんじゃないの？

- `nil : forall X : Type, list X` が型を引数とする関数なのと同様
- `b_sum : forall (n m : nat), ...` は自然数を引数とする関数
  - どちらも引数によって型が変わる
  - `nil nat : list nat`
  - `nil bool : list bool`
  - `b_sum 3 5 : beautiful 3 -> ...`
  - `b_sum 4 9 : beautiful 4 -> ...`

今日のメニュー

Prop.v

- 帰納的に定義される命題
- 証明オブジェクト
  - 証明スクリプトと証明オブジェクト
  - 含意 (「ならば」) と関数
  - 証明オブジェクトに関する帰納法
  - 偶数性
  - 計算による定義と帰納的定義
  - 証拠についての場合わけ・inversion
- ...

証明オブジェクトと証明スクリプト

- **証明オブジェクト**: 命題を型として見た時に，その型を持つ式
- **証明スクリプト**: `Proof.` と `Qed.` の間に書いてあるもの
  - **タクティック**: 証明オブジェクトを (少しずつ) 作るための命令
  - `Show Proof`: **構成途中の証明オブジェクトを表示するコマンド**
    - `Proof.` の後が穴 (`?1`, `?2`) の空いた証明オブジェクト
    - `Subgoals` の下に各穴に埋められるべき証明オブジェクトの型—つまり証明すべきサブゴール
    - (ここの `->` は「ならば」や「関数」とは無関係)
    - サブゴールが無くなると完全な式 (証明) になる

証明オブジェクトも式である

証明オブジェクトを直接書くなら `Theorem` **コマンドすら不要**

```
Definition eight_is_beautiful''' : beautiful 8
  b_sum 3 5 b_3 b_5.
```

４通りの証明の中身の比較:

```
Print eight_is_beautiful.
Print eight_is_beautiful'.
Print eight_is_beautiful''.
Print eight_is_beautiful'''.
```

今日のメニュー

Prop.v

- **帰納的に定義される命題**
- **証明オブジェクト**
  - **証明スクリプトと証明オブジェクト**
  - **含意 (「ならば」) と関数**
  - **証明オブジェクトに関する帰納法**
  - **偶数性**
  - **計算による定義と帰納的定義**
  - **証拠についての場合わけ・inversion**
- ...

含意(「ならば」)と関数 (1/2)

```
Theorem b_plus3: forall n,
  beautiful n -> beautiful (3+n).
Proof.
   intros n H.  apply b_sum.
   apply b_3.  apply H.
Qed.
```

- この定理の証明オブジェクトは？
- より一般に命題 P->Q の証明オブジェクトとは？
- -> は関数の型なので…

含意(「ならば」)と関数 (2/2)

P->Q の証明オブジェクトは

P の証明オブジェクトを引数として
Q の証明オブジェクトを返す関数!

```
Definition b_plus3' :
  forall n, beautiful n -> beautiful (3+n) :=
    fun n => fun H : beautiful n =>
      b_sum 3 n b_3 H.
```

$$\frac{\vdots}{\texttt{beautiful n}} \quad \mapsto \quad \frac{\dfrac{}{\texttt{beautiful 3}}\,\mathrm{B3} \qquad \dfrac{\vdots}{\texttt{beautiful n}}}{\texttt{beautiful (3+n)}}\,\mathrm{BSUM}$$

より関数定義っぽく書いた証明

```
Definition
  b_plus3'' (n : nat) (H : beautiful n)
    : beautiful (3+n) :=
  b_sum 3 n b_3 H.
```

宿題： 12/5 午前10:00 締切

- Exercise: `six_is_beautiful` (1), `nine_is_beautiful` (1), `b_times2` (2), `b_timesm` (2)
- **解答を書き込んだ `Prop.v` をまるごとオンライン提出システムを通じて提出**
- **以下をコメント欄に明記**:
  - **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．(「特になし」はダメです．)**
  - **友達に教えてもらったら、その人の名前，他の資料 (web など) を参考にした場合，その情報源 (URL など)．**