「計算と論理」
Software Foundations
その7

五十嵐 淳

cal15@fos.kuis.kyoto-u.ac.jp

http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/cal/

December 22, 2015

Prop.v

帰納的に定義される命題

- 偶数性と偶数に関する帰納法
- 「美しい」数と証明オブジェクトについての帰納法
- 証明オブジェクトについての inversion
- その他の例
- (帰納的に定義される) 関係

命題を返す関数を使った偶数性の定義

```
Coq < Definition even(n:nat) : Prop :=
Coq <   evenb n = true.
even is defined

Coq < Check even.
even
     : nat -> Prop
```

```
Theorem two_is_even : even 2.
Proof.
  unfold even.  reflexivity.
Qed.
```

(この方法では命題として使える基本的な語彙は増えていない.)

偶数の集合の帰納的定義

偶数の集合 EV は，以下のふたつの条件を満たす最小の (自然数の部分) 集合である.

- `O` は EV の元である
- ***n*** が EV の元ならば `S`（`S` ***n***）は EV の元である

- 「ふたつの条件」だけを満たす[1]集合はいくらでもある
  - 例えば 1 以外の自然数の集合
- 最小性で「ゴミ」を取り除く

$\Longrightarrow$ 帰納的定義 $=$ 規則について閉じている $+$ 最小性

[1] 「規則について閉じている」ともいう

述語「偶数性」の帰納的定義

集合を述語と思うと「述語の帰納的定義」になる

自然数 n が偶数である ($ev\ n$ と書く) とは，以下の規則から帰納的に定義される.

- $ev\ \mathtt{O}$ である
- 任意の自然数 n について，$ev\ n$ ならば $ev\ (\mathtt{S}\ (\mathtt{S}\ n))$ である

自然演繹流で書けば

新しい原子命題 ev n:

$$\frac{\Gamma \vdash n : \mathtt{nat}}{\Gamma \vdash \mathtt{ev}\ n : \mathtt{Prop}} \quad \text{(Ev-P)}$$

- 導入規則:

$$\frac{}{\Gamma \vdash \mathtt{ev}\ \mathtt{O}} \quad \text{(Ev-I1)}$$

$$\frac{\Gamma \vdash \mathtt{ev}\ n}{\Gamma \vdash \mathtt{ev}\ (\mathtt{S}\ (\mathtt{S}\ n))} \quad \text{(Ev-I2)}$$

Coq での帰納的述語の定義

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

- 新しい (自然数を引数とする) 命題 ***ev*** の定義
  - `nat -> Prop` ⇒ 自然数に関する述語
- コンストラクタ ≒ 導入規則の名前
  - あたかも定理のように使える
- コンストラクタの型 ≒ 規則の内容
  - `forall n:nat` ⇒ 規則のパラメータ
  - `->` ⇒ 規則の前提と結論の間の水平線

例: 「4 は偶数である」

```
Coq < Check ev_0.
ev_0
     : ev 0

Coq < Check (ev_SS 0).
ev_SS 0
     : ev 0 -> ev 2

Coq < Check (ev_SS 0 ev_0).
ev_SS 0 ev_0
     : ev 2

Coq < Check (ev_SS 2 (ev_SS 0 ev_0)).
ev_SS 2 (ev_SS 0 ev_0)
     : ev 4
```

例: 「4 は偶数である」

```
Theorem four_is_even : even 4.
Proof.
  apply ev_SS.  apply ev_SS.  apply ev_0.
Qed.

Print four_is_even.
```

- four_is_even (の中身) を even 4 の証明オブジェクト (教科書では evidence とも) という
  - even 4 型の項，といってもよい
  - even 4 の導出木に対応

例

```
Theorem double_even : forall n:nat,
  even (double n).
Proof.



Qed.
```

偶数に関する帰納法

$P(n)$ を自然数 n の性質について述べた命題とする

偶数に関する帰納法の原理

「任意の偶数 n について $P(n)$」は以下と同値

- $P(0)$ かつ
- 任意の偶数 n' について $P(n')$ ならば $P(S(S\ n'))$

- ふたつの場合分けは偶数性の定義に対応

自然演繹流で書くと

$$\frac{\Gamma \vdash \mathsf{ev}\ \ m \qquad \Gamma \vdash P[O] \qquad \Gamma, n : nat, IH : P[n] \vdash P[S(Sn)]}{\Gamma \vdash P[m]}\ (\text{Ev-E})$$

証明の例

evenb 関数 (Basics.v 参照) についての性質

定理: 自然数 *n* が偶数ならば，*evenb n* は `true` を返す

証明: 偶数に関する帰納法.

- $n = 0$ の場合. *evenb* 0 は明らかに `true` を返す
- $n = S(S\ n')$ かつ n' が偶数の場合.
  evenb $(S(S\ n'))$ を定義に従い計算すると，
  evenb n' と等しい．一方，帰納法の仮定より n' については *evenb* n' が `true` を返すので，*evenb n* も `true` を返す.

(証明終)

Coq での証明

```
Definition even (n:nat) : Prop :=
  evenb n = true.

Theorem ev__even : forall n, ev n -> even n.
Proof.
  intros n E. induction E as [| n' E'].
  - (* E = ev_0 *).
    unfold even. reflexivity.
  - (* E = ev_SS n' E' *)
    unfold even. simpl. apply IHE'.
Qed.
```

induction E って?

- $E : ev\ n$ ということは,
  - $E = \texttt{ev\_0}$ かつ $n = 0$
  - $E = \texttt{ev\_SS}\ n'\ E'$ かつ $E' : ev\ n'$ (つまり n' も偶数)

  のいずれか.
- E は (枝分かれしない導出木で) ある種のリスト構造をしている!

$\Longrightarrow$ `induction E` はリストに関する帰納法のようなもの!

- 「偶数性の導出に関する帰納法」ともいう

同じ証明の別の書き方

定理: 自然数 n が偶数ならば，*evenb n* は `true` を返す

証明: `ev` n の導出に関する帰納法．最後の規則について場合分け．

- `ev_0` の場合．この時 $n = 0$．*evenb* 0 は明らかに `true` を返す
- `ev_SS` の場合，この時ある n' について $n = S(S\ n')$ かつ `ev` n' である．*evenb* $(S(S\ n'))$ を定義に従い計算すると，*evenb* n' と等しい．一方，帰納法の仮定より n' については *evenb* n' が `true` を返すので，*evenb n* も `true` を返す．(証明終)

“= true” vs. Inductive による命題定義

Q: evenb があるのに，なぜ ev を定義するの？

A1: 偶数に関する帰納法が使える

A2: 一般的に,「○○性」の (規則による) 定義が書けても，何かが「○○性」を満たすかを判定する関数が (簡単に) 書けるとは限らない

Prop.v

帰納的に定義される命題

- 偶数性と偶数に関する帰納法
- 「美しい」数と証明オブジェクトについての帰納法
- その他の例
- 証明オブジェクトについての inversion
- (帰納的に定義される) 関係

「美しい」自然数 (深い意味は特にない)

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:

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

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

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

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

「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

述語 `beautiful n` で “n is beautiful” を表す.

```
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).
```

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.
```

```
Theorem beautiful_plus_eight: forall n,
  beautiful n -> beautiful (8+n).
Proof.
  intros n B.
  apply b_sum with (n:=8) (m:=n).
  apply eight_is_beautiful.
  apply B.
Qed.
```

証明オブジェクトに関する帰納法

- `Inductive` による型定義
  →その型のデータに関する帰納法
  - 「任意の自然数 n について…」の証明
- `Inductive` による命題定義
  →その命題の証明に関する帰納法
  - 「`beautiful n` ならば…」の証明
  - 「任意の `beautiful n` の証明オブジェクト E について…」の証明

「n は美しい数である」の導出

beautiful n の証明オブジェクト E について，以下のいずれかがいえる:

- $E = \texttt{b\_0}$ かつ $n = 0$
- $E = \texttt{b\_3}$ かつ $n = 3$
- $E = \texttt{b\_5}$ かつ $n = 5$
- $E = \texttt{b\_sum}\ n1\ n2\ E1\ E2$ かつ
  - $n = n1+n2$ かつ
  - $E1$ は beautiful $n1$ の (E より小さい) 証明オブジェクト，かつ，
  - $E2$ は beautiful $n2$ の (E より小さい) 証明オブジェクト

beautiful n の導出に関する帰納法，または美しい数に関する帰納法

「任意の beautiful n なる n について $P(n)$」つまり「任意の n について beautiful n ならば $P(n)$」と以下は同値

- $P(0)$ かつ
- $P(3)$ かつ
- $P(5)$ かつ
- 任意の $n1, n2$ について $P(n1)$ かつ $P(n2)$ ならば $P(n1 + n2)$

導出の形に関する場合分けとの対応!

自然演繹流で書くと

$$\frac{\Gamma \vdash \texttt{beautiful}\ e \qquad \Gamma \vdash P[O] \qquad \Gamma \vdash P[3] \qquad \Gamma \vdash P[5] \qquad \Gamma, m : nat, n : nat, IHb1 : P[m], IHb2 : P[n] \vdash P[m+n]}{\Gamma \vdash P[e]}\ (\textsc{Beautiful-E})$$

または,

$$\frac{\Gamma \vdash P[O] \qquad \Gamma \vdash P[3] \qquad \Gamma \vdash P[5] \qquad \Gamma, m : nat, n : nat, IHb1 : P[m], IHb2 : P[n] \vdash P[m+n]}{\Gamma \vdash \forall k : nat, \texttt{beautiful}\ k \texttt{ -> } P[k]}\ (\textsc{Beautiful-E'})$$

例: 帰納的定義される命題の同値性証明(1)

```
Inductive gorgeous : nat -> Prop :=
  g_0 : gorgeous 0
| g_plus3 :
   forall n, gorgeous n -> gorgeous (3+n)
| g_plus5 :
   forall n, gorgeous n -> gorgeous (5+n).
```

明らかに(?), ゴージャスさと美しさは同値

$\Longrightarrow$ 証明してみよう!

- その前に…

ゴージャス数の導出

gorgeous n の導出 E について，以下のいずれかがいえる:

- $E = \texttt{g\_0}$ かつ $n = 0$
- $E = \texttt{g\_plus3}\ n1\ E1$ かつ $n = 3 + n1$ かつ $E1$ は gorgeous $n1$ の (E より小さい) 導出
- $E = \texttt{g\_plus5}\ n1\ E1$ かつ $n = 5 + n1$ かつ $E1$ は gorgeous $n1$ の (E より小さい) 導出

ゴージャスな数についての帰納法

gorgeous n の導出に関する帰納法，またはゴージャスな数に関する帰納法

「任意の gorgeous n なる n について $P(n)$」つまり「任意の n について gorgeous n ならば $P(n)$」と以下は同値

- $P(0)$ かつ
- 任意の $n1$ について $P(n1)$ ならば $P(3 + n1)$
- 任意の $n1$ について $P(n1)$ ならば $P(5 + n1)$

自然演繹流で書くと

$$\frac{\Gamma \vdash \texttt{gorgeous}\ e \qquad \Gamma \vdash P[O] \qquad \Gamma, n : nat, IHg : P[n] \vdash P[3 + n] \qquad \Gamma, n : nat, IHg : P[n] \vdash P[5 + n]}{\Gamma \vdash P[e]}$$

(Gorgeous-E)

または,

$$\frac{\Gamma \vdash P[O] \qquad \Gamma, n : nat, IHg : P[n] \vdash P[3 + n] \qquad \Gamma, n : nat, IHg : P[n] \vdash P[5 + n]}{\Gamma \vdash \forall k : nat, \texttt{beautiful}\ k \texttt{ -> } P[k]}$$

(Gorgeous-E')

ゴージャスな自然数は美しい!

```
Theorem gorgeous__beautiful : forall n,
  gorgeous n -> beautiful n.
Proof.
   intros n E.
   induction E as [| n' | n' ].
     (* 導出に関する帰納法! *)
   - (* g_0 *)  apply b_0.
   - (* g_plus3 *)
     apply b_sum. apply b_3. apply IHgorgeous.
   - (* g_plus5 *)
     apply b_sum. apply b_5. apply IHgorgeous.
Qed.
```

美しい自然数はゴージャス!

```
Theorem gorgeous_sum : forall n m,
  gorgeous n -> gorgeous m -> gorgeous (n + m).

Theorem beautiful__gorgeous : forall n,
  beautiful n -> gorgeous n.
```

Prop.v

帰納的に定義される命題

- 偶数性と偶数に関する帰納法
- 「美しい」数と証明オブジェクトについての帰納法
- 証明オブジェクトについての inversion
- その他の例
- (帰納的に定義される) 関係

証明オブジェクトについての inversion

偶数性の導出を使った推論:

- 帰納法 (induction)
- 導出を「遡る」 (inversion)

例題:

```
Theorem ev_minus2: forall n,
  ev n -> ev (pred (pred n)).
Theorem SSev_ev : forall n,
  ev (S (S n)) -> ev n.
```

$n = 0$ と n が2以上の偶数の場合で場合分け

```
Theorem ev_minus2: forall n,
  ev n -> ev (pred (pred n)).
Proof.
  intros n E.
  inversion E as [| n' E'].
  - (* E = ev_0 *) simpl. apply ev_0.
  - (* E = ev_SS n' E' *) simpl. apply E'.
Qed.
```

destruct だと失敗する

```
Theorem SSev_ev_firsttry : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  inversion E as [| n'].
    (* The goal is still "ev n" *)
```

inversion だとうまくいく

しかも，ありえない場合を取り除いてくれる!

```
Theorem SSev__even : forall n,
  ev (S (S n)) -> ev n.
Proof.
  intros n E.
  inversion E as [| n' E'].
    (* E = ev_0 なわけがない! *)
    (* E = ev_SS n' E' *)
    apply E'.
Qed.
```

任意の自然数について *ev (S(S n))* ならば *ev n*

(証明) ***ev (S(S n))*** の導出について場合分け

- ev_0 の場合: ありえない.
- ev_SS の場合: 導出の前提は ***ev n*** のはずなので題意が示せる.

(証明終)

導出に対する inversion

文脈で H : I (I は帰納的に定義された命題) とする時, inversion H は:

- コンストラクタ (導出規則) 毎に場合わけ
  - ev_0, ev_SS の場合
- 各場合での前提条件…
  - …を文脈に追加
    - ★ ev_SS の場合の前提 ev n が追加
  - …が矛盾している場合は場合そのものの除去
    - ★ ev_0 の場合 (S (S n) = 0 はありえない)

を一気に行う

Prop.v

帰納的に定義される命題

- 偶数性と偶数に関する帰納法
- 「美しい」数と証明オブジェクトについての帰納法
- 証明オブジェクトについての inversion
- その他の例
- (帰納的に定義される) 関係

リストに対する述語

ev_list l: 「*l* は偶数長リストである」

```
Inductive ev_list {X:Type} : list X -> Prop :=
  | el_nil : ev_list []
  | el_cc : forall x y l,
              ev_list l -> ev_list (x :: y :: l).
```

- 空リストは偶数長リストである
- *l* が偶数長リストならば *x* :: *y* :: *l* は偶数長リストである

偶数長リストの長さは偶数である

```
Lemma ev_list__ev_length:
  forall X (l : list X),
    ev_list l -> ev (length l).
Proof.
    intros X l H. induction H.
    - (* el_nil *)
      simpl. apply ev_0.
    - (* el_cc *)
      simpl.  apply ev_SS. apply IHev_list.
Qed.
```

Prop.v

帰納的に定義される命題

- 偶数性と偶数に関する帰納法
- 「美しい」数と証明オブジェクトについての帰納法
- 証明オブジェクトについての inversion
- その他の例
- (帰納的に定義される) 関係

命題としての関係

- 一引数の命題 (一引数述語): 「もの」の性質を表す
  - `beautiful`, `even` など
- 二引数の命題 (二引数述語): 「もの」と「もの」の関係を表す
  - `eq`

関係「以下」の帰納的定義

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

導出規則:

$$\frac{}{n \leq n} \quad (\text{LE-N})$$

$$\frac{n \leq m}{n \leq \mathrm{S}\ m} \quad (\text{LE-S})$$

「以下」に関する証明

基本的には ev, beautiful などと同じ:

- ゴールにあるならコンストラクタを apply
- 文脈にあるなら inversion

```
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.
```

「未満」の定義

```
Definition lt (n m:nat) := le (S n) m.
Notation "m < n" := (lt m n).
```

- `le` を使わずに直接帰納的な定義をするとしたら？

宿題： / 午前10:30 締切

- Exercise: `b_times2` (2), `gorgeous_plus13` (1), `gorgeous_sum` (2), `ev_sum` (2), `inversion_practice` (1)
- 解答を書き込んだ `Prop.v` までのファイルを全て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).