「計算と論理」
Software Foundations
その8

五十嵐 淳
cal15@fos.kuis.kyoto-u.ac.jp

京都大学

January 5, 2015

本日のメニュー

MoreLogic.v (続・Coq の論理)

- 特称量化 (「ある x が存在して〜」)
- 命題と真偽値の融合

特称量化

「型 X の要素 x が存在して P」($\exists x : X.P$) の定義:

```
Inductive ex (X:Type) (P : X->Prop) : Prop :=
  ex_intro : forall (witness:X),
                P witness -> ex X P.

Notation "'exists' x , p" := (ex _ (fun x => p))
  (at level 200, x ident, right associativity)
```

- 直観: `ex X P` の証拠は型 X の要素 *witness* と P *witness* の証拠の組
  - 存在証明には「何が」存在するかを実際に示す

自然演繹における導出規則

$$\frac{\Gamma, x : T \vdash P : \mathtt{Prop}}{\Gamma \vdash \exists x : T, P : \mathtt{Prop}} \quad (\mathrm{P}\text{-}\exists)$$

$$\frac{\Gamma \vdash e : T \qquad \Gamma \vdash P[e]}{\Gamma \vdash \exists x : T, P[x]} \quad (\exists\text{-}\mathrm{I})$$

$$\frac{\Gamma \vdash \exists x : T, P[x] \qquad \Gamma, x : T, H : P[x] \vdash Q \qquad \Gamma \vdash Q : \mathtt{Prop}}{\Gamma \vdash Q} \quad (\exists\text{-}\mathrm{E})$$

特称量化に関する証明(1)

witness を指定して apply ex_intro を使う

```
Example exists_example_1 :
  exists n, n + (n * n) = 6.
Proof.
  apply ex_intro with (witness:=2).
  reflexivity.  Qed.
```

- apply ex_intro with (withness:= ***e***) の代わりに exists ***e*** でも OK

特称量化に関する証明(2)

文脈に特称量化がある時は destruct (か inversion) を使う

```
Theorem exists_example_2 : forall n,
  (exists m, n = 4 + m) ->
  (exists o, n = 2 + o).
Proof.
  intros n H.  destruct H as [m Hm].
    (* witness に intro パターンで名前をつける
    (* m が実際に何かはわからないが
       Hm : n = 4 + m は満たす *)
  exists (2 + m).
  apply Hm.  Qed.
```

直観主義論理と特称量化

- Coq の論理では $\exists x, \neg P$ と $\neg(\forall x, P)$ は同値ではない! (dist_not_exists, not_exists_dist 参照)
  - 前者ならば後者，のみ成立
    - ★ 古典論理ではどちらも同値
  - $\forall x, \neg P$ と $\neg(\exists x, P)$ は同値
  - $\forall x, P$ と $\neg(\exists x, \neg P)$ ではどうか？
  - $\exists x, P$ と $\neg(\forall x, \neg P)$ ではどうか？
- その他
  - $(\forall x, (P \vee \neg P))$ -> $(\exists x, P) \vee (\forall x, \neg P)$ も成り立ちそうで (？) 成り立たない
    - ★ 古典論理では成立

本日のメニュー

MoreLogic.v (続・Coq の論理)

- 特称量化 (「ある x が存在して〜」)
- 命題と真偽値の融合

Prop vs. bool 再び

- Prop…証明・推論に使うが，計算には使えない
  - 例えば，`if a = b then ... else ...` はダメ
- 等しさ，偶数性などを計算に使いたかったら:
  - `beq_nat`, `evenb` などの関数を定義
  - 命題との関係を証明する
    - ★ 例:

```
Theorem beq_nat_true : forall n m,
    beq_nat n m = true -> n = m.
```

$\Longrightarrow$ 面倒!

sumbool 型

```
Inductive sumbool (A B : Prop) : Type :=
 | left : A -> sumbool A B
 | right : B -> sumbool A B.

Notation "{ A } + { B }" :=  (sumbool A B) : typ
```

- $\{A\} + \{B\}$ の値は
  - *left a* (ただし *a* は命題 *A* の証明オブジェクト)
  - *right b* (ただし *b* は命題 *B* の証明オブジェクト)
- ほぼ *or* の定義と同じ
- *Type* だと計算に使える!(後述)
  - (教科書では *Type* ではなく *Set* になっているが, 気にしない)

```
Theorem eq_nat_dec : forall n m : nat,
  {n = m} + {n <> m}.
Proof. (* 教科書参照 *) Defined.
```

- dec は decidable の略
- + は，ふつうの or だと思って証明すればよい．
- eq_nat_dec の返値型は Prop ではなく Type なので関数として計算に使える!
- 最後を Qed. でなく Defined. でしめると，定義内容が見える (計算中に展開できる) ようになる

eq_nat_decの使用例

```
Definition override' {X: Type}
  (f: nat->X) (k:nat) (x:X) : nat->X:=
    fun (k':nat) => if eq_nat_dec k k' then x
                    else f k'.

Theorem override_same' :
  forall (X:Type) x1 k1 k2 (f : nat->X),
    f k1 = x1 ->
    (override' f k1 x1) k2 = f k2.
```

証明

```
Proof.
  intros X x1 k1 k2 f. intros Hx1.
  unfold override'.
  destruct (eq_nat_dec k1 k2).
  (* observe what appears as a hypothesis *)
  Case "k1 = k2".
    rewrite <- e. symmetry. apply Hx1.
  Case "k1 <> k2".
    reflexivity.  Qed.
```

BasicTactics.v での証明[1]と比べてみよう

[1]講義ではやってない

証明オブジェクト

eq_nat_dec の証明オブジェクトは,

- 自然数 n, m を受け取る (全域) 関数
- 返値は
  - n と m が等しければ $left\ (n = m)(\sim n = m)a$ (ただし，a は $n = m$ の証明オブジェクト)
  - そうでなければ $right\ (n = m)(\sim n = m)b$ (ただし，b は $n <> m$ の証明オブジェクト)

プログラム抽出

- 証明オブジェクト = 関数定義 (計算の記述) と証明の合成物
- プログラム抽出: 証明オブジェクトから計算に関わる部分を取り出す

自然数 n, m を受け取り，n と m が等しければ ***left*** を，そうでなければ ***right*** を返す関数

- `Extraction eq_nat_dec.` とすると自然数の等価判定を行う (OCaml) プログラムが表示される!

$\Longrightarrow$ 正しさが Coq で保証されたプログラム!

宿題： / 午前10:30 締切

- Exercise: `dist_not_exists` (1), `dist_exists_or` (2)
- 解答を書き込んだ `MoreLogic.v` までのファイルを全てを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).