Coqゼミ　第3回

浅井研究室（担当：廣田知子）

2008年6月30日

1　関数の拡張

従来の関数は

```
fun 引数:型 => 式
```

という形であった。引数の型をA、式の型をBとすれば、この関数の型は$A \to B$であった。

今までは暗黙の了解としてA, Bは型だと思っていたが、式と型を区別しないのであれば、AやBは型だけでなくソートにしても良い。(A, B)がそれぞれ型かソートかによって、以下の四通りが考えられる。

（型、型） 今までの普通の関数。

（型、ソート）

式を受け取ったら、型を返す関数。

型が値に依存するので、この関数の型を**依存型**という。

（ソート、型）

型を受け取ったら式を返す関数。

典型的には、多相関数がこれに当たる。

（ソート、ソート）

型を受け取ったら、型を返す関数。例えばlistなど。

2　$\Pi x : A.\, B$

A型のxを受け取ったら、B型のものを返す関数の型を表す。通常の関数の型$A \to B$と違うのは、xがB中に現れても良いという点。（xがB中に現れていなければ、$\Pi x : A.\, B$と$A \to B$は同じものである。）

また、

$\Pi x : *.\, B$

は、任意の型のxを受け取ったら、B型のものを返す関数の型を表す。つまり、$\forall A.B$という多相型のこと。（例えば、$\Pi A : *.\, A \to A$とすれば、これは任意の型のAについて$A \to A$つまり、ほぼ 'a -> 'a型の関数のこと。）

3　Coqでの$\Pi x : A.B$の表記法

Coqでは、$\Pi x : A.B$をforall x:A, Bと書く。

以下の三式は全く同じ意味を表す：

```
forall x:A, forall y:A, forall z:B, M
forall x y:A, forall z:B, M
forall (x y:A) (z:B), M
```

4　例1（program）

例えば以下のprogramを考えてみる。

```
Section id_section.
  Variable A:Set.
  Definition f : A -> A := fun n:A => n.
End id_section.
```

関数fは、任意の型Aを受け取ってきたら、fun n : A => n（A型の値nを受け取ったらそのままnを返す関数）を返す。つまりfは多相恒等関数である。

```
Coq < Print f.
f = fun (A : Set) (n : A) => n
  : forall A : Set, A -> A

Coq < Eval compute in (f nat).
 = fun n:nat => n
 : nat -> nat

Coq < Eval compute in (f nat 3).
 = 3 : nat
```

5　例2（proof term）

```
Coq < Print id_p.
id_p = fun (P : Prop) (H : P) => H
     : forall P : Prop, P -> P
```

Prop を Set にすれば、前節の関数 f と全く同じ式。論理学的には、任意の命題 P について「P ならば P」が成り立つという意味、つまり述語論理の $\forall P.\, P \to P$ の意味を持つ。計算論的には、任意の型 P を受け取ってきたら `fun H : P => H` という関数を返す。この関数は、P 型の値 H を受け取ったら、それをそのまま返す。「P 型の値 H」は「命題 P の証明 H」と読めるので、それを受け取ったら、その証明をそのまま返す、という意味にもなる。

6 述語論理

とあるデータ n について、P という性質が成り立つ、というのを P(n) などと書く。例えば $n > 0$ など。これは、データを受け取ったら真偽値を返す命題となる。つまり P は「型 → ソート」型の関数であり、依存型を持つ。

例えば、Coq ではあらかじめ `le_n` と `le_S` という自然数上の述語が用意されている。

```
Coq < Check le_n.
le_n : forall n:nat, n <= n

Coq < Check le_S.
le_S : forall n m:nat,
       n <= m -> n <= S m
```

任意の自然数 n について、前者は $n \leq n$ を示す述語、後者は $n \leq m$ なら $n \leq m+1$ であることを示す述語である。

7 $36 \leq 38$ の証明

(1) le_n より、36≤36 である。(apply le_n)
(2) (1) と le_S より、36≤37 である。
(apply le_S)
(3) (1),(2) と le_S より、36≤38 である。
(apply le_S)

これを逆順に Coq に入れると証明出来る。

```
Coq < Print le_36_38.
le_36_38 = le_S 36 37 (le_S 36 36
                       (le_n 36))
         : 36 <= 38
```

8 $n \leq n+2$ の証明

前節と同様。ただし、頭に「任意の n について（forall n:nat,）」を付ける。forall は関数の型とほとんど同じなので、intro を使える。

9 蛇足 1（コマンド）

ctl-C-N
　一行読み込む。（next ボタンに相当）
ctl-C-U
　一行戻る。（undo ボタンに相当）
ctl-C-return
　ポインタの置いてある行まで読み込む。（goto ボタンに相当）
ctl-K
　ポインタの置いてある行を消去する。

ProofGeneral 上では上記のコマンドを使うことが出来ます。覚えておくと便利です。

10 蛇足 2（コメント文の書き方）

`(* コメント文 *)` と書きます。

練習問題

問 1
$n \leq n+2$ の証明を自分でやってみよ。

問 2
以下を証明せよ。証明完成後、証明を表す式を *Print* コマンドで見てみよ。

```
(1) Lemma all_perm :
     forall (A : Prop) (p : A -> A -> Prop),
     (forall x y : A, P x y) ->
      forall x y : A, P y x.

(2) Lemma all_imp_dist :
     forall (A : Prop) (P Q : A -> Prop),
     (forall a : A, P a -> Q a) ->
     (forall a : A, P a) ->
      forall a : A, Q a.

(3) Lemma all_delta :
     forall (A : Prop) (R : A -> A -> Prop),
     (forall a b : A, R a b) ->
      forall a : A, R a a.
```