# Coqゼミ　第一回

浅井研究室（担当：廣田知子）

2008年6月19日

## 1　Coqとは？

Coqとは定理証明系システムの一つです。（他にもisabelleなどがあります。）

定理証明系システムとは、名前の通りコンピュータプログラムを使って数学的定理を証明するシステムのことです。

## 2　Coqのインストール

Coqのインストール方法は浅井研のWikiにのっているので、そちらを参考にして下さい。（アドレスはhttp://dellsv/wikiです。）ちなみにCoqのホームページはhttp://coq.inria.fr/です。

## 3　Coqの起動

ターミナルからcoqtopとすると起動します。あるいはCoqIDEを入れているなら、それを起動すれば良いです。

Coqファイルを保存するときは拡張子を.vにして下さい。

## 4　宣言等

Coqのコマンドは全て．（ピリオド）で終わります。

同じ行に続けてコマンドを打つ場合は、必ずコマンド同士の間をスペースで空けて下さい。

Require Import Arith.
　自然数の四則演算をするときに書く。

Require Import ZArith.
　整数の四則演算をするときに書く。

Require Import Bool.
　真偽値を使うときに書く。

Require Import List.
　リストを使うときに書く。

## 5　基本的なコマンド

Check 式.　　式を型チェックする。

Print 式.　　式の値を表示する。

## 6　基本的な値

自然数 0, S。型は nat。

真偽値 true, false。型は bool。

Unset Printing Notations.
　とすると、自然数が0とSを使った表記で表示される。

Set Printing Notations.
　で自然数を使った表記になる。

## 7　名前無し関数の定義

```
fun 変数 : 型 => 式.
```

以下の三つは同じ意味：

```
fun a:nat => fun b:nat => fun c:bool => 式.
fun a b:nat => fun c:bool => 式.
fun (a b:nat) (c:bool) => 式.
```

## 8　大域的変数定義（Definitionコマンド）

**Definition**コマンドで定数も非再帰的関数も定義出来る。

Definition 変数 : 型 := 式.
（自明な場合は : 型 を省略出来る）

例：

```
Definition x : nat := 3.
Definition cube : nat -> nat :=
                    fun n:nat => n * n * n.
Definition cube (n:nat) : nat := n * n * n.
Definition cube n : nat := n * n * n.
```

## 9　ブロック構造

Section セクション名.
　本体.
　本体.
　...
End セクション名.

そのブロックの中でのみ有効な局所変数定義を宣言出来る。上記の本体部分には、大域的変数定義の他、局所変数定義を使える。

## 10 局所変数定義

Variable 変数名 : 型.

一度に複数個宣言するときは、Variables を用いる。

Variables 変数名 : 型.

例：

```
Section sum_section.
 Variables a b : nat.
 Definition sum := a + b.
End sum_section.
```

ブロックを閉じた後、sum を関数として使えるようになる。局所変数は定義されたブロックの中でしか使用出来ないことに注意。

## 11 実行

Eval compute in 式.

例：

```
Eval compute in 3+4.
Eval compute in (sum 1 4).
```

## 12 式：型：ソート

**program : specification : Set**

specification
: 型（ソート）が Set であるような式。
例： nat → nat

program
: 型が specification であるような式。
例： fun n:nat ⇒ n.

## 13 Specification の定義

大域的変数定義と同じ。

```
Definition nat_bin : Set := nat -> nat -> nat.
```

## 14 Realization

その型を持つようなプログラムを作ること。例えば、型 $nat \to nat \to nat \to nat$ の realization は以下のように書ける。

```
Section g_section.
  Variables a b c : nat.
  Definition g := a * b + c.
End g_section.
```

このプログラムは以下のように書いても同じ：

```
Definition g a b c := a * b + c.
```

その型の realization は一通りでなく、複数通り存在する可能性があることに注意。

## 練習問題

**問 1**

上に出てくる式を全て入力し、何が起きるか確かめよ。又、定義された式について、*Check* と *Print* を行い、型と値を確かめよ。

**問 2**

型 $(nat \to bool) \to (bool \to nat) \to (nat \to nat)$ の *realization* を示せ。それを *coq* の上で実行して確かめよ。