Jacques Garrigue, 2017 年 11 月 29 日

数学的な証明

1 $\sqrt{2}$ が無理数である

まずは自然数で以下の定理を証明する。

定理 1 任意の自然数 n と p について、

$$n \cdot n = 2(p \cdot p) \text{ ならば } p = 0$$

証明は n の関する整礎帰納法を使う。

- $n = 0$ のとき、$p = 0$
- $n \neq 0$ のとき、
  - n と p が偶数でなければならないので、$n = 2n', p = 2p'$ とおける
  - 再び、$n' \cdot n' = 2(p' \cdot p')$ が得られ、$n' < n$
  - 帰納法の仮定より $p' = 0$
  - すなわち、$p = 0$

その定理を使って、$\sqrt{2}$ が無理数であることを証明する。もしも $\sqrt{2}$ が有理数なら、ある n と p が存在し、$\sqrt{2} = n/p$、すなわち $n^2 = 2p^2$。しかし上の定理から $p = 0$ となるので矛盾。

整礎帰納法の原理

前回は整礎帰納法による関数の定義を見た。当然ながら、整礎帰納法は定理の証明にも使える。

```
Require Import Wf_nat.
Check lt_wf_ind.
     : forall (n : nat) (P : nat -> Prop),
       (forall n0 : nat, (forall m : nat, m < n0 -> P m) -> P n0) -> P n

Goal forall n, n + 0 = n.
  induction n using lt_wf_ind.            (* induction ... using ... で使う *)
  destruct n.
    reflexivity.
  simpl.
  f_equal.
  apply H.
  auto.
Qed.
```

Coq の実数

Coq には `Reals` という実数の標準ライブラリがある。自然数 `nat` や整数 `Z` と違い、実数の型 `R` は公理で定義されている。実数を使って証明はできるが、計算はできない。

本定理の証明

```
Require Import Arith Wf_nat Omega.

(* 半分についての様々な補題 *)

Module Div2.
Definition double n := n + n.
Fixpoint div2 (n : nat) :=
  match n with
  | 0 | 1 => 0
  | S(S n') => S (div2 n')
  end.

Check plus_n_Sm.

Parameter double_div2: forall n, div2 (double n) = n.

Parameter double_inv: forall n m, double n = double m ->  n = m.

Theorem double_mult_l: forall n m,  double (n * m) = double n * m.
  unfold double. auto with arith.
Qed.

Theorem double_mult_r: forall n m,  double (n * m) = n * double m.
  unfold double; intros; ring.
Qed.

Lemma div2_le : forall n, div2 n <= div2 (S n) <= S (div2 n).
  induction n. split. auto. auto.                (* 実は /\ でつながっている *)
  destruct IHn.
Admitted.

Lemma div2_lt : forall n, n <> 0 -> div2 n < n.
Proof.
  induction n; intros.
    elim H; auto.
  destruct n; simpl.
    auto.
  destruct (div2_le n).
  apply lt_n_S.
  eauto using le_lt_trans.
Qed.
End Div2.
Import Div2.

(* 偶数について *)

Require Import Even.                              (* 標準ライブラリの Even を使う *)
Print even.                    (* 授業で見たものと違い、相互再帰で定義されている *)
Inductive even : nat -> Prop :=
  | even_0 : even 0
```

```
  | even_S : forall n : nat, odd n -> even (S n)
  with odd : nat -> Prop :=
  | odd_S : forall n : nat, even n -> odd (S n)

Theorem even_is_even_times_even: forall n, even (n * n) ->  even n.
  intros.
  destruct (even_or_odd n). auto.
  eapply even_mult_inv_r; eauto.
Qed.

Parameter double_even : forall n, even (double n).

(* 相互帰納法の原理を生成する *)
Scheme even_odd_ind := Induction for even Sort Prop
  with odd_even_ind := Induction for odd Sort Prop.

Check even_odd_ind.                    (* even と odd の両方に対して述語を取る *)
     : forall (P : forall n : nat, even n -> Prop)
              (P0 : forall n : nat, odd n -> Prop),
       P 0 even_0 ->
       (forall (n : nat) (o : odd n), P0 n o -> P (S n) (even_S n o)) ->
       (forall (n : nat) (e : even n), P n e -> P0 (S n) (odd_S n e)) ->
       forall (n : nat) (e : even n), P n e

Lemma even_double : forall n, even n -> double (div2 n) = n.
Proof.
  eapply even_odd_ind.                          (* odd の述語がまだ分からない *)
      reflexivity.
    intros.
    apply H.                                  (* odd について仮定をそのまま返す *)
Admitted.                                              (* 証明を完成させてね *)

(* 本定理で使う補題 *)
Theorem even_square: forall n,
  even n ->  double (double (div2 n * div2 n)) = n * n.
Admitted.

(* 本定理 *)
Theorem main_thm: forall n p : nat, n * n = double (p * p) ->  p = 0.
Proof.
  induction n using lt_wf_ind; intros.                  (* 整礎帰納法の使い方 *)
  destruct (eq_nat_dec n 0).       (* 自然数なら、比較について排中律が成り立つ *)
    subst.
    destruct p; try discriminate.
    auto.
  assert (even_n: even n). admit.                        (* ここも完成させてね *)
  assert (even_p: even p). admit.
  rewrite <- (even_double p even_p).
  rewrite <- (even_double _ even_0).
  f_equal.                                                 (* 両辺の関数を取る *)
  apply (H (div2 n)).
Admitted.
```

無理数であることの証明

```
(* 実数の世界に移る *)

Require Import Reals.
Require Import Field.                    (* 体における単純化をする作戦 field *)

Print Raxioms.
Check completeness.                  (* 空でない上に有界な集合には上限がある *)
     : forall E : R -> Prop,
       bound E -> (exists x : R, E x) -> {m : R | is_lub E m}

(* 無理数の定義 *)

Definition irrational (x : R) : Prop :=
   forall (p : Z) (q : nat), q <> 0 ->  x <> (IZR p / INR q)%R.

(* sqrt 2 が無理数である *)

Theorem irrational_sqrt_2: irrational (sqrt (INR 2)).
Proof.
  intros p q Hq Hrt.
  elim Hq.
  Check Zabs_nat.
  apply (main_thm (Zabs_nat p)).
  replace (Div2.double (q * q)) with (2 * (q * q))
    by (unfold Div2.double; ring).
  destruct (eq_nat_dec (Zabs_nat p * Zabs_nat p) (2 * (q * q))).
    assumption.
  elim (not_nm_INR _ _ n).                          (* 実数の等式に変える *)
  repeat rewrite mult_INR.
  Check sqrt_def.
  rewrite <- (sqrt_def (INR 2)) by auto with real.
  rewrite Hrt.
  assert (INR q <> 0%R). auto with real.
  destruct p; simpl; try field; auto.
Qed.
```

未完成な証明

まだ証明していない事実を証明したかのように扱うには何通りのやり方もある。

- Parameter (または Axiom などの同義語) で公理として加える。
- 証明の途中で Admitted で無理矢理証明を認めさせる。これで Qed で終われたように見える。
- 証明の途中で admit で現在のゴールを認めさせる。次のゴールに移る。

練習問題 1.1 証明の中の Parameter を Theorem に、Admitted を Qed に変え、admit をなくして、証明を完成せよ。