MathComp/SSReflect 2

Jacques Garrigue, 2018 年 12 月 12 日

4　MathComp による証明

数学関係の定理

こちらはモジュールが多過ぎて、簡単に紹介できない。よく使うものとして、割り算 div、総和などの集合に対する演算子 bigop、有限集合 finset(fintype に基く)、基本的な線形代数は matrix、perm や vector、多項式は poly、素数は prime などがある。

2 の平方根が無理数

実数は MathComp のものではないが、それ以外は MathComp でできる。

```
From mathcomp Require Import all_ssreflect.

Module Irrational.
Require Import Wf_nat Reals Field.

Lemma odd_square n : odd n = odd (n*n).
Proof. by rewrite odd_mul andbb. Qed.

Lemma even_double_half n : ~~odd n -> n./2.*2 = n.
Proof. by rewrite -[RHS]odd_double_half => /negbTE ->. Qed.

(* 本定理 *)
Theorem main_thm (n p : nat) : n * n = (p * p).*2 -> p = 0.
Proof.
  elim/lt_wf_ind: n p => n.
  case: (posnP n) => [-> _ [] // | Hn IH p Hnp].
  have Hn2 : (n./2 < n)%coq_nat.
  apply/ltP; by rewrite -divn2 ltn_Pdiv.
  have even_n : ~~odd n by rewrite odd_square Hnp odd_double.
  move: Hnp; rewrite -(even_double_half n) //.
  rewrite -muln2 mulnAC mulnA !muln2 => /double_inj Hnp.
  have even_p : ~~odd p by rewrite odd_square -Hnp odd_double.
  move: Hnp; rewrite -(even_double_half p) // => /esym.
  by rewrite -muln2 mulnAC mulnA !muln2 => /double_inj /esym /IH ->.
Qed.

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

Theorem irrational_sqrt_2: irrational (sqrt (INR 2)).
```

```
Proof.
  move=> p q Hq Hrt.
  elim Hq.
  apply (main_thm (Zabs_nat p)).
  apply INR_eq.
  rewrite -mul2n !mult_INR -(sqrt_def (INR 2)) ?Hrt; last by auto with real.
  have Hqr : INR q <> 0%R by auto with real.
  case: p => /= [|p|p]. by field.
    rewrite INR_IPR /IZR; by field.
  rewrite INR_IPR /IZR; by field.
Qed.
End Irrational.
```

京都大学 2009 年度の入試問題

ほとんど sssrnat と div で証明できる.

ここでは、初歩数論の div を応用した例をみる。大石君の提案で 2009 年度の京大の入試問題の一部である。

> a、b を互いに素な正の整数とし、a が奇数である。そのときに、$(a+b\sqrt{2})^n = a_n + b_n\sqrt{2}$ と定める。
>
> 1. a_2 は奇数であり、a_2 と b_2 は互いに素である。
> 2. 全ての n について、a_n は奇数であり、a_n と b_n は互いに素である。

とりあえず、(1) の一部と (2) のヒントを与える。

```
Record ext2 : Set := Ext2 {pnat: nat; pirr: nat}.      (* 構造体の定義 *)

Definition prod2 (a b : ext2) :=
  Ext2 (pnat a*pnat b+2*pirr a*pirr b) (pnat a * pirr b + pirr a * pnat b).

Section Problem1.
Variables a b : nat.

Hypothesis odd_a : odd a.
Hypothesis Hab : coprime a b.

Definition ab := Ext2 a b.

Theorem odd_a2 : odd (pnat (prod2 ab ab)).
Proof.
  rewrite /prod2 /=.
  by rewrite odd_add odd_mul -mulnA mul2n odd_double addbC odd_a.
Qed.

Theorem mpr_a2 : coprime (pnat (prod2 ab ab)) (pirr (prod2 ab ab)).
Proof.
  rewrite /=.
  rewrite [b*a]mulnC addnn -muln2.
```

```
  rewrite !coprime_mulr /coprime.
  rewrite (gcdnC _ a) gcdnMDl.
  rewrite -/(coprime a (2 * b * b)) !coprime_mulr !Hab.
  rewrite coprimen2 odd_a /=.
Admitted.
End Problem1.

Section Problem2.
Lemma odd_coprime2 ab n :
  odd (pnat ab) -> coprime (pnat ab) (pirr ab) ->
  let ab2n := iter n (fun ab => prod2 ab ab) ab in
  odd (pnat ab2n) && coprime (pnat ab2n) (pirr ab2n).
Admitted.

Theorem odd_coprime ab n :
  odd (pnat ab) -> coprime (pnat ab) (pirr ab) ->
  let abn := iter n (prod2 ab) ab in
  odd (pnat abn) && coprime (pnat abn) (pirr abn).
Admitted.
End Problem2.
```

名古屋大学 2013 年度の入試問題

k, m, n は自然数で、$k \geq 0$, $m \geq 2$, $n \geq 1$ とする。

$$S_k(n) = \sum_{i=1}^{n} i^k \qquad T_m(n) = \sum_{k=1}^{m-1} {}_mC_k S_k(n)$$

1. $T_m(1)$ と $T_m(2)$ を求めよ。
2. 一般的な n に対して、$T_m(n)$ を求めよ。
3. p が 3 以上の素数のとき、$S_k(p-1)$ $(1 \leq k \leq p-2)$ は p の倍数であることを証明せよ。

```
Section Nagoya2013.
Definition Sk k n := \sum_(1<=i<n.+1) i^k.

Variable m : nat.
Hypothesis Hm : m > 1.

Definition Tm n := \sum_(1<=k<m) 'C(m,k) * Sk k n.

Lemma Sk1 k : Sk k 1 = 1.
Proof. by rewrite /Sk big_nat1 exp1n. Qed.

Lemma Tm1 : Tm 1 = 2^m - 2.
Proof.
  rewrite /Tm.
  rewrite [in 2^m](_ : 2 = 1+1) //.
```

```
  rewrite Pascal.
  transitivity ((\sum_(0 <= k < m.+1) 'C(m,k)) - 2).
    symmetry.
    rewrite (@big_cat_nat _ _ _ m) //=.
    rewrite (@big_cat_nat _ _ _ 1) //=; last by apply ltnW.
    rewrite addnAC !big_nat1.
    rewrite bin0 binn addKn.
    apply eq_bigr => i _.
    by rewrite Sk1 muln1.
  rewrite big_mkord.
  congr subn.
  apply eq_bigr => i _.
  by rewrite !exp1n !muln1.
Qed.

Lemma Tm2 : Tm 2 = 3^m - 3.
Proof.
  rewrite /Tm.
  have ->: 3^m - 3 = 2^m - 2 + (3^m - 1 - 2^m).
    rewrite addnC addnBA.
      rewrite subnK.
        by rewrite -subnDA.
      rewrite subn1 -ltnS prednK.
        by apply ltn_exp2r, ltnW.
      by rewrite expn_gt0.
    admit.
  rewrite -Tm1.
  rewrite [in 3^m](_ : 3 = 1+2) //.
  rewrite Pascal.
  transitivity (Tm 1 + (\sum_(1 <= k < m) 'C(m,k) * 2^k)).
    admit.
  congr addn.
  transitivity ((\sum_(0 <= k < m.+1) 'C(m,k) * 2^k) - 1 - 2^m).
Admitted.

Theorem Tmn n : Tm n.+1 = n.+2^m - n.+2.
Proof.
  elim:n => [|n IHn] /=.
    by apply Tm1.
Admitted.

Theorem Skp p k : p > 2 -> prime p -> 1 <= k < p.-1 -> p %| Sk k p.-1.
Admitted.
End Nagoya2013.
```

練習問題 4.1 上記の証明の admit と Admitted をなくせ。