

SSReflect 3

Jacques Garrigue, 2015 年 12 月 9 日

4 前回の課題

Section Coq10.

Variables (A : Type) (P Q : A -> bool).

```
Lemma remove_c : forall (a : A),
  (forall x y, Q x = Q y) ->
  (forall c, ((exists x, P x) -> P c) -> Q c) -> Q a.
```

Proof.

```
move=> a Eq H.
case Ha: (Q a).
  done.
rewrite -Ha.
apply H.
move=> [b pb].
move/negbT/negP: Ha.
elim.
rewrite (Eq a b).
by apply H.
Qed.
```

End Coq10.

Section Problem1.

....

Lemma coprimeMD1 k c d : coprime c (k * c + d) = coprime c d.

Proof. by rewrite /coprime gcdnMD1. Qed.

Theorem mpr_a2 : coprime (pnat (prod2 ab ab)) (pirr (prod2 ab ab)).

Proof.

```
rewrite /=.
rewrite [b*a]mulnC addnn -muln2.
rewrite !coprime_mulr.
rewrite coprime_sym coprimeMD1.
rewrite !coprime_mulr !Hab.
rewrite coprime2 odd_a /=.
rewrite addnC coprime_sym coprimeMD1.
rewrite !coprime_mulr coprime_sym !Hab /=.
rewrite -mulnA mulnC coprime_sym coprimeMD1.
by rewrite coprime_mulr !coprime2n odd_a.
```

Qed.

End Problem1.

5 課題の続き

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).
Proof.
  elim: n => [|n IHn] odd_a copr_ab.
Admitted.

Lemma prod2A a b c : prod2 a (prod2 b c) = prod2 (prod2 a b) c.
Proof.
  rewrite /prod2.
  destruct a, b, c => /=.
  f_equal; ring.
Qed.

Lemma prod2e a : prod2 a (Ext2 1 0) = a.
Admitted.

Lemma prod2_iter n ab x :
  iter n (prod2 ab) x = prod2 (iter n (prod2 ab) (Ext2 1 0)) x.
Admitted.

Lemma prod2_2n n ab :
  iter n (fun ab => prod2 ab ab) ab =
  iter (2^n) (prod2 ab) (Ext2 1 0).
Admitted.

Lemma prod2_2n_Sn n ab :
  iter n (fun ab => prod2 ab ab) ab =
  iter (2^n - n.+1) (prod2 ab) (iter n (prod2 ab) ab).
Proof.
  have Hn : 2^n - n.+1 + n.+1 = 2^n.
  by rewrite subnK // ltn_expl.
Admitted.

Lemma not_odd_coprime ab ab' n :
  odd (pnat ab) -> coprime (pnat ab) (pirr ab) ->
  odd (pnat ab') && coprime (pnat ab') (pirr ab') = false ->
  let abn := iter n (prod2 ab) ab' in
  odd (pnat abn) && coprime (pnat abn) (pirr abn) = false.
Proof.
  move=> odd_a copr_ab.
  elim: n ab' => // n IHn ab' Hab'.
  move: IHn(IHn (prod2 ab ab')).
  rewrite iterSr.
  apply.
  case odd_a': (odd (pnat ab')) Hab'.
Admitted.

Lemma 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).
Proof.
  move=> odd_a copr_ab /=.

```

```

set abn := iter n _ ..
case Hneg: ( _ && _ ) => //.
move: (not_odd_coprime _ _ (2^n - n.+1) odd_a copr_ab Hneg).
rewrite -prod2_2n_Sn => <-.
by apply odd_coprime2.
Qed.
End Problem2.

```

6 TPPmark 2014 の問題

定理証明支援系の集まりで解いた問題。

http://imi.kyushu-u.ac.jp/lasm/tpp2014/index_ja.html

練習問題 6.1 \mathbb{N} を自然数の集合とし、 $p \in \mathbb{N}$, $q \in \mathbb{N}$ とする。 $p \bmod q = r$ は $p = kq + r$ となるような自然数 k の存在を表す。 $(p \mid q)$ は $(p \bmod q) = 0$ を意味する。そのときに以下の 3 つを証明せよ。

1. 任意の $a \in \mathbb{N}$ について、 $(a^2 \bmod 3) = 0$ または $(a^2 \bmod 3) = 1$.
2. 任意の自然数 a, b, c について、 $a^2 + b^2 = 3c^2$ ならば $(3 \mid a)$ かつ $(3 \mid b)$ かつ $(3 \mid c)$.
3. 任意の自然数 a, b, c について、 $a^2 + b^2 = 3c^2$ ならば $a = b = c = 0$.

7 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。

$$S_k(n) = \sum_{i=1}^n i^k \quad T_m(n) = \sum_{k=1}^{m-1} {}_m C_k S_k(n)$$

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。

```

Require Import ssreflect ssrbool ssrnat ssrfun choice eqtype seq fintype.
Require Import finfun bigop binomial div prime. (* MathComp *)

```

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 expln. 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 7.1 上記の証明の admit と Admitted をなくせ。