代数的拡張

Jacques Garrigue, 2022 年 1 月 26 日

1　De Bruijn 添字について

解は講義のホームページにあるので、ここには重要な定義だけを載せる。

λ 項　　$M ::= n \mid \lambda M \mid MM$

β 簡約　　$(\lambda M)N \to [N/0]M$

代入

$$[N/k]n \;=\; \begin{cases} N & n = k \\ n & n < k \\ n-1 & n > k \end{cases} \qquad \begin{aligned} [N/k]\lambda M &= \lambda[\Uparrow_0 N/k+1]M \\ [N/k](M_1\ M_2) &= [N/k]M_1\ [N/k]M_2 \end{aligned}$$

シフト

$$\Uparrow_k n = \begin{cases} n+1 & k \le n \\ n & k > n \end{cases} \qquad \Uparrow_k \lambda M = \lambda \Uparrow_{k+1} M \qquad \Uparrow_k (M_1\ M_2) = \Uparrow_k M_1\ \ \Uparrow_k M_2$$

簡約関係

$$\frac{M \to M'}{M\ N \to M'\ N} \qquad \frac{N \to N'}{M\ N \to M\ N'} \qquad \frac{M \to M'}{\lambda M \to \lambda M'} \qquad M \Rightarrow M \qquad \frac{M \to M' \quad M' \Rightarrow N}{M \Rightarrow N}$$

2　帰納法で追い越して戻る

2009 年度の京都大学理学部の入試問題を問いてみよう。

> 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。

(2) の解き方のポイントは、この性質は 2 乗で保存されるが、簡単に n 乗から $n+1$ 乗に移せさい。その代り、$n+1$ 乗から n 乗に戻ることができる。

また、試験では代数的拡大が有利分と無理分に分解できることを基地とするが、せっかくなので、それも問題に加えた。

> 0. $f(a,b) = a + b\sqrt{2}$ および $(a_1,b_1) \times (a_2,b_2) = (a_1a_2 + 2b_1b_2, a_1b_2 + b_1a_2)$ とする $(a, b, \ldots$ は自然数$)$. 以下を証明せよ.
>
> (a) f は単射である
>
> (b) $f((a_1,b_1) \times (a_2,b_2)) = f(a_1,b_1)f(a_2b_2)$

コードの中に多くの補題を書いたが、重要なのは各節の最後の定理だけなので、途中の補題を自由に変えていい。

```
From mathcomp Require Import all_ssreflect all_algebra.

Definition ext2 : Set :=  nat * nat.                              (* a + b√2 *)
Definition pnat (x : ext2) := x.1.
Definition pirr (x : ext2) := x.2.

Definition prod2 (a b : ext2) : ext2 :=                          (* ext2 上の積 *)
  (pnat a * pnat b + 2 * pirr a * pirr b,
   pnat a * pirr b + pirr a * pnat b).

Section Problem1.                                                (* まず問 1 から *)
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 /=.
Search _ odd addn.
Admitted.

Lemma coprimeMDl k c d : coprime c (k * c + d) = coprime c d.
Proof. by rewrite /coprime gcdnMDl. Qed.

(* 問 1 の証明 *)
Theorem mpr_a2 : coprime (pnat (prod2 ab ab)) (pirr (prod2 ab ab)).
Proof.
rewrite /=.
rewrite [b*a]mulnC addnn -muln2.
rewrite !coprimeMr coprime_sym coprimeMDl.
rewrite !coprimeMr !Hab.
rewrite coprimen2 odd_a /=.
Admitted.                                                        (* 残りをよろしく *)
End Problem1.

Section Problem2.                                    (* 問 2 が難しいのでヒント多め *)
Definition prod22 ab := prod2 ab ab.

(* (a + b√2)^(2^n) の整数部が奇数で，整数部と無理数部が互いに素 *)
Lemma odd_coprime2 ab n :
  odd (pnat ab) -> coprime (pnat ab) (pirr ab) ->
  let ab2n := iter n prod22 ab in
  odd (pnat ab2n) && coprime (pnat ab2n) (pirr ab2n).
Proof.
elim: n => [|n IHn] odd_a copr_ab /=.
Admitted.

Lemma prod2e : right_id (1,0) prod2.
Proof. rewrite /prod2 => -[a b] /=. f_equal; ring. Qed.
```

```
Lemma prod2A : associative prod2.
Admitted.

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

About iterD.   iter (n + m) f x = iter n f (iter m f x)
Lemma prod2_2n n ab : iter n prod22 ab = iter (2^n) (prod2 ab) (1,0).
Admitted.

About iterSr.  iter n.+1 f x = iter n f (f x)
Lemma prod2_2n_Sn n ab :
  iter n prod22 ab = iter (2^n - n.+1) (prod2 ab) (iter n (prod2 ab) ab).
Admitted.

About dvdnP.      reflect (exists k : nat, m = k * d) (d %| m)
About dvdn_gcdl.  gcdn m n %| m
About dvdn_gcdl.  gcdn m n %| n
About gcdnMl.     gcdn n (m * n) = n
About oddM.       odd (m * n) = odd m && odd n

(* (a + b√2)^(m+n) について成り立てば, (a + b√2)^m についても成り立つ *)
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.

(* 問2の証明 *)
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).
Proof.
move=> odd_a copr_ab /=.
set abn := iter n _ _.
case Hneg: (_ && _) => //.
Admitted.
End Problem2.

Section Problem0.                              (* prod2 の定義の根拠を確認する *)
Import GRing.Theory Num.Theory.
Open Scope ring_scope.
```

```
Variable R : rcfType.                          (* Real closed field: 実数体の型 *)

(* 上の定義を実数上に解釈する *)
Definition eval (x : ext2) : R := (pnat x)%:R + (pirr x)%:R * Num.sqrt 2%:R.

(* √2 が無理数だと仮定する（第5回に近いものを証明した）*)
Hypothesis irrat2 : forall p q : int, Num.sqrt 2%:R <> p%:~R / q%:~R :> R.

Lemma natr_inj : injective (fun x => (x%:R : R)).
Proof. move=> x y /mulrIn -> //; exact: oner_neq0. Qed.

(* ext2 と実数上での表現の間の同型：単射と準同型 *)

About mulrzBl_nat. ∀[M:zmodType](m n:nat) (x:M), x *~ (m - n) = x *~ m - x *~ n
About unitfE.      ∀[F:fieldType](x:F), (x \is a GRing.unit) = (x != 0)
About subr_eq0.    ∀[V:zmodType](x y:V), (x - y == 0) = (x == y)
(* 解釈が単射 *)
Theorem eval_inj : injective eval.
Proof.
rewrite /eval => -[x1 x2] [y1 y2] /=.
move/(f_equal (fun x : R => x - x2%:R * Num.sqrt 2%:R)).
rewrite addrK -addrA -mulNr -mulrDl.
case/boolP: (x2 == y2) => [/eqP -> | neq].
Admitted.

About expr2.       ∀[R:ringType](x:R), x ^+ 2 = x * x
About sqr_sqrtr.   ∀[R:rcfType][a:R], 0 <= a -> Num.sqrt a ^+ 2 = a
About natrM.       ∀(R:ringType)(m n:nat), (m * n)%:R = m%:R * n%:R
About natrD.       ∀(R:ringType)(m n:nat), (m + n)%:R = m%:R + n%:R
(* 解釈が積の準同型 *)
Theorem prod2_morph : {morph eval : x y / prod2 x y >-> x * y}.
Proof.
rewrite /prod2 /eval => x y /=.
Admitted.
End Problem0.
```

練習問題 2.1 上記の証明の admit と Admitted をなくせ.