線形合同法の周期

TAoCP 3.2.1.2 の形式化

坂口和彦

筑波大学 情報学群 情報科学類 B3

2013/11/21

目標

線形合同法が最大周期を持つ条件の正当性を Coq(SSReflect) で証明する

`https://github.com/itplanning/lcg-ssreflect`

目標

線形合同法が最大周期を持つ条件の正当性を Coq(SSReflect) で証明する

https://github.com/itplanning/lcg-ssreflect

SSReflect とは?

- Coq の拡張+ライブラリ集
- 代数関連のライブラリが充実している
- 多くの命題を `Prop` ではなく `bool` で書けるように工夫されている
- リフレクションで `bool` と `Prop` を読み替えられる

線形合同数列

$$
\begin{array}{ll}
m & (0 < m) \\
a & (0 \leq a < m) \\
c & (0 \leq c < m) \\
X_0 & (0 \leq X_0 < m)
\end{array}
$$

に関して

$$X_{n+1} = (aX_n + c) \mod m$$

で定義される数列を線形合同数列と呼ぶ

```
Variable (cM' : nat).
Definition cM := cM'.+1.
Variable (cA cC : 'I_cM).
```

```
Definition nextr (x : 'I_cM) : 'I_cM :=
  @Ordinal cM ((cA * x + cC) %% cM)
           (ltn_pmod (cA * x + cC) (ltn0Sn cM')).
```

最大周期

X_0 から X_{m-1} が全て違う数でかつ $X_m = X_0$ であるとき、線形合同数列が最大周期を持つ

```
Definition full_period n :=
  uniq (rseq cM n) && (iter cM nextr n == n).
```

最大周期の条件

線形合同数列が最大周期を持つことは

- c と m が互いに素
- $a-1$ が m の持つ全ての素因数で割り切れる
- m が 4 の倍数ならば、$a-1$ も 4 の倍数

を全て満たすことと同値である

```
Definition full_period' :=
  [&& coprime cM cC,
      all (fun p => cA %% p == 1) (primes cM) &
      if 4 %| cM then cA %% 4 == 1 else true].
```

この同値性の証明が、今回の目標

歴史

- T. E. Hull and A. R. Dobell (1962) Random Number Generators, SIAM Review, Vol. 4, No. 3, pp.230-254
  片方向だけの証明が載っている
- D. E. Knuth. The Art of Computer Programming, Volume 2: Seminumerical Algorithms, Third Edition. Addison-Wesley, 1997.
  両方向の証明が載っている

X_0 はなんでも良い

- 最大周期を持つ線形合同数列は、X_0 を別の数にしても最大周期を持つ
- $X_0 = 0$ の場合だけを考える

線形合同数列の一般項 - 1

以下の 2 つは同値な定義:

$$X_0 = 0$$
$$X_{n+1} = (aX_n + c) \mod m$$

$$X_n \equiv c \sum_{k=0}^{n-1} a^k \mod m$$

入れ子になっている mod を外すには:

```
eqn_modDr (p m n d : nat) :
  (m + p == n + p %[mod d]) = (m == n %[mod d])
modnMmr (m n d : nat) : m * (n %% d) = m * n %[mod d]
```

線形合同数列の一般項 - 2

$$X_n \equiv c \sum_{k=0}^{n-1} a^k \mod m$$

X_n は明らかに c と m の最大公約数の倍数なので、最大周期になるのは c と m が互いに素である場合だけ

互いに素であれば c をどう取っても最大周期であるかどうかは変わらないので、$c = 1$ の場合だけを考える

線形合同数列の一般項 - 2

$$X_n \equiv c \sum_{k=0}^{n-1} a^k \mod m$$

X_n は明らかに c と m の最大公約数の倍数なので、最大周期になるのは c と m が互いに素である場合だけ
互いに素であれば c をどう取っても最大周期であるかどうかは変わらないので、$c = 1$ の場合だけを考える

線形合同数列の一般項 - 3

$$X_n \equiv \sum_{k=0}^{n-1} a^k \mod m$$

総和の部分は`\sum_(k < n) cA ^ k`と書けるが、多くの場合は`iter n (fun x => (x * cA).+1) 0`の方が便利

良く知られた性質:

$$\sum_{k=0}^{n-1} a^k = \frac{a^n - 1}{a - 1}$$

線形合同数列の一般項 - 3

$$X_n \equiv \sum_{k=0}^{n-1} a^k \mod m$$

総和の部分は`\sum_(k < n) cA ^ k`と書けるが、多くの場合は`iter n (fun x => (x * cA).+1) 0`の方が便利

良く知られた性質:

$$\sum_{k=0}^{n-1} a^k = \frac{a^n - 1}{a - 1}$$

線形合同数列の一般項 - 3

$$X_n \equiv \sum_{k=0}^{n-1} a^k \mod m$$

総和の部分は`\sum_(k < n) cA ^ k`と書けるが、多くの場合は`iter n (fun x => (x * cA).+1) 0`の方が便利

良く知られた性質:

$$(a-1)\sum_{k=0}^{n-1} a^k = a^n - 1$$

TAoCP 3.2.1.2

- Theorem A (目的の定理)
  - Lemma P
  - Lemma Q
  - Lemma R

Lemma P

$$\forall p \in \mathbb{P}, x \in \mathbb{N}, e > 0 \ .$$
$$2 < p^e \ \wedge \ p^e \mid x - 1 \ \wedge \ p^{e+1} \nmid x - 1 \implies$$
$$p^{e+1} \mid x^p - 1 \ \wedge \ p^{e+2} \nmid x^p - 1$$

SSReflect では $p^n \mid x \wedge p^{n+1} \nmid x$ となる n を、`logn p x` と書ける (ただし `logn p 0 = 0`)

```
forall p x. prime p -> 2 < p ^ logn p x ->
  logn p (x.+1 ^ p).-1 = (logn p x).+1
```

証明のほとんどは書き換えと場合分けだけでできる、19 行
二項定理 (`binomial.Pascal`) を使う

Lemma P

$$\forall p \in \mathbb{P}, x \in \mathbb{N}, e > 0 \ .$$
$$2 < p^e \ \wedge \ p^e \mid x-1 \ \wedge \ p^{e+1} \nmid x-1 \implies$$
$$p^{e+1} \mid x^p - 1 \ \wedge \ p^{e+2} \nmid x^p - 1$$

SSReflect では $p^n \mid x \wedge p^{n+1} \nmid x$ となる n を、`logn p x` と書ける (ただし `logn p 0 = 0`)

```
forall p x. prime p -> 2 < p ^ logn p x ->
  logn p (x.+1 ^ p).-1 = (logn p x).+1
```

証明のほとんどは書き換えと場合分けだけでできる、19 行
二項定理 (`binomial.Pascal`) を使う

Lemma P

$$\forall p \in \mathbb{P}, x \in \mathbb{N}, e > 0 \ .$$
$$2 < p^e \ \wedge \ p^e \mid x - 1 \ \wedge \ p^{e+1} \nmid x - 1 \implies$$
$$p^{e+1} \mid x^p - 1 \ \wedge \ p^{e+2} \nmid x^p - 1$$

SSReflect では $p^n \mid x \wedge p^{n+1} \nmid x$ となる n を、`logn p x` と書ける (ただし `logn p 0 = 0`)

```
forall p x. prime p -> 2 < p ^ logn p x ->
  logn p (x.+1 ^ p).-1 = (logn p x).+1
```

証明のほとんどは書き換えと場合分けだけでできる、19 行

二項定理 (`binomial.Pascal`) を使う

Lemma Q

m の素因数分解が $m = {p_0}^{e_0} \ldots {p_{t-1}}^{e_{t-1}}$ であり、任意の $0 \leq p < t$ について $a' = a \bmod {p_j}^{e_j}, m' = {p_j}^{e_j}$ による線形合同数列が最大周期を持つならば、a, m による線形合同数列は最大周期を持つ

Lemma R

$p \in \mathbb{P}, 1 < a < p^e$ ならば、$\sum_{k=0}^{\lambda-1} a^k \equiv 0 \pmod{p^e}$ を満たす最小の正整数 λ に関して

$$\lambda = p^e \Leftrightarrow \begin{cases} a \equiv 1 \pmod{p} & (2 < p) \\ a \equiv 1 \pmod{4} & (p = 2) \end{cases}$$

が成り立つ

```
forall p x e l,
let zero i := p ^ e %| iter i (fun a => (a * x).+1) 0 in
prime p -> 1 < x < p ^ e -> 0 < l ->
(forall l', 0 < l' < l -> ~~ zero l') -> zero l ->
(l == p ^ e) = ((if p == 2 then 4 else p) %| x.-1).
```

λ を周期だと思って読むと理解しやすい

Lemma R

$p \in \mathbb{P}, 1 < a < p^e$ ならば、$\sum_{k=0}^{\lambda-1} a^k \equiv 0 \pmod{p^e}$ を満たす最小の正整数 λ に関して

$$\lambda = p^e \Leftrightarrow \begin{cases} a \equiv 1 \pmod{p} & (2 < p) \\ a \equiv 1 \pmod{4} & (p = 2) \end{cases}$$

が成り立つ

```
forall p x e l,
let zero i := p ^ e %| iter i (fun a => (a * x).+1) 0 in
prime p -> 1 < x < p ^ e -> 0 < l ->
(forall l', 0 < l' < l -> ~~ zero l') -> zero l ->
(l == p ^ e) = ((if p == 2 then 4 else p) %| x.-1).
```

λ を周期だと思って読むと理解しやすい

フェルマーの小定理

Lemma R の証明の中でフェルマーの小定理が出てくるが

$$\forall p \in \mathbb{P}, x \in \mathbb{N}.p \mid x^p - x$$

という形をしている

さらに、実際にフェルマーの小定理から導かれた (と書いてある) 命題は

$$p \mid a^{p^e} - a$$

と書いてあり、辻褄を合わせるのが難しい

フェルマーの小定理

Lemma R の証明の中でフェルマーの小定理が出てくるが

$$\forall p \in \mathbb{P}, x \in \mathbb{N}. p \mid x^p - x$$

という形をしている

さらに、実際にフェルマーの小定理から導かれた (と書いてある) 命題は

$$p \mid a^{p^e} - a$$

と書いてあり、辻褄を合わせるのが難しい

苦労した点

- TAoCP の証明は結構大雑把に書かれている
  - Coq で書き直すには情報量が足りない
  - Lemma R の証明の一部分で練習問題を参照している部分も
- $p = 2$ とそれ以外の場合で証明が全く違う部分が多い
- TAoCP の日本語版には $a^{2^{e-1}}$ がどう見ても a^{2e-1} になっている部分があり、混乱した

得られた知見 - 1

- SSReflect のライブラリには初等整数論の問題を解くのに必要なものが揃っている
- `0 < n`, `1 < n`, `prime n` 等が出てきたときに、まず `n` で場合分けすると証明全体が簡潔に記述できる
  - `prednK n : 0 < n -> n.-1.+1 = n` での rewrite を減らせる

得られた知見 - 2

- rewrite やリフレクションを使って証明を進める限りは証明可能だったゴールが証明不能になることがない
  - SSReflect での証明のほとんどの部分は rewrite で書けるようになっている
- SSReflect では Coq の標準ライブラリには無い notation や coercion が多数使われていて、実際の term が何なのか分からなくなりやすい
  - `Set Printing All.` すると、実際の term が見える
  - `n < m` の定義が `n.+1 <= m` になっていることを忘れやすい

今後について

年末出す本 (同人誌) に記事を載せる予定

http://tcug.jp/ (Tsukuba Coq Users' Group)