SSReflectによる符号理論の形式化に関する一考察

アフェルト　レナルド

産業技術総合研究所(AIST)
情報セキュリティ研究センター(RCIS)

動機と背景

- 証明可能な安全性証明の形式化の研究成果は複数ある
  - ほぼすべて**整数論**に基づく（ElGamal, BBS, RSA-OAEP等）
- 今のところ,**符号**ベース暗号の形式化は困難
  - 形式化された代数ライブラリが整備されていないため
- マイクロソフト・INRIAで,代数の形式化を進めている
  - GonthierらはFeit-Thompson定理の形式化に向けて

⇒ このライブラリは符号ベース暗号の証明可能安全性の証明の形式化に適用できるのではないか

今回の発表

- SSReflectについてのサーベイ
  - マイクロソフト・INRIAが開発中の代数のライブラリ
  - 定理証明支援器Coq [INRIA, 1985~]に基づく
    - Coqは高階論理の実装
    - 非常に低レベル
    - まとまった数学ライブラリはまだ整備されていない
- 例として符号理論を利用するが, 簡単な定義や結果しか扱わない

- 本発表ではCoqのコードをそのまま見せる
  - 最近の改良によって, だいぶ読みやすくなった
  - この発表ではフォントや色を多少変更した

Coq (ASCII)	本発表
/¥	∧
->	→
forall	∀
exists	∃
!=	≠
¥in	∈
=>	⇒
<->	↔

概要

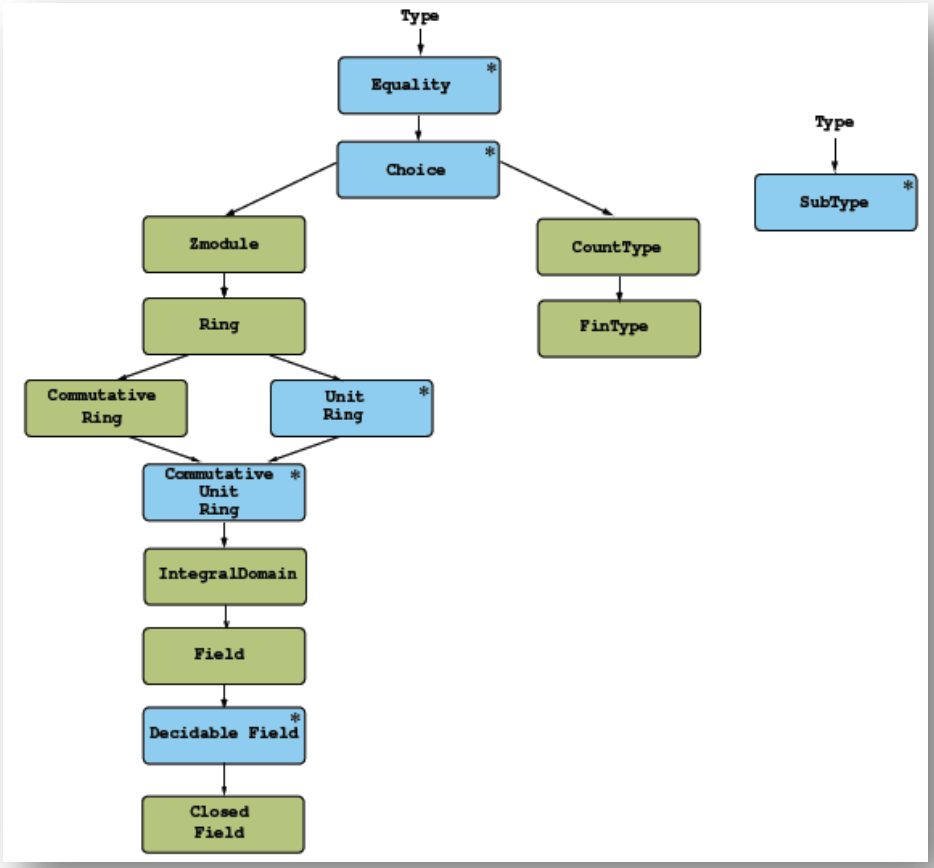
- SSR の基礎
  - ssralgの階層
  - 基礎的なデータ構造
  - 例 : ハミング重み・距離
- 線形符号を定義してみよう
  - 線形代数から得られる性質
  - 最近傍復号についての簡単な定理
- McElieceを定義してみよう
  - ゴッパ符号を仮定・乱数生成は無視
- 結論

ssralgライブラリの構造

- 代数構造の階層
  - 例：可換環による配列の環による多項式の環
- 例：素数pを法とする有限体
  - SSReflectの構文：

```
'F_p
```

  - 体の性質だけでなく，環の性質も持つ



[F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau, "Packaging mathematical structures", TPHOLs 2009]

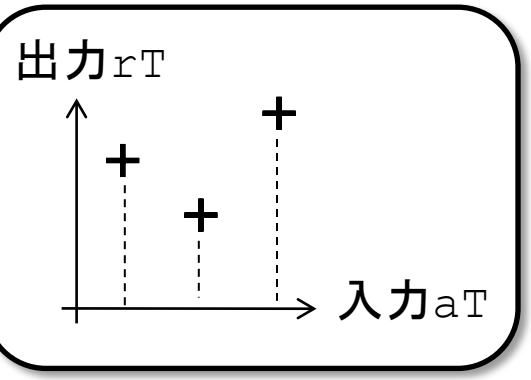
有限ドメインの関数

- 問題：Coqのネイティブな関数は一般的過ぎる
  - 数学の関数よりアルゴリズムに近い
    - 同じ入力・出力の関係を満たす関数は等しい？
      - クイックソート関数 ↔ マージソート関数
- アイデア：関数はグラフとして形式化する

`{ffun aT → rT}`：関数（aTの要素は有限）

```
Inductive finfun_type : predArgType :=
  Finfun of #|aT|.-tuple rT.
```

リストの長さ＝要素の数



- 例:

```
fun x ⇒ ~~ x: bool → bool

[ffun x ⇒ ~~ x] : {ffun bool → bool}

fgraph [ffun x ⇒ ~~ x] : #|bool_finType|.-tuple bool

Goal [ffun x ⇒ ~~ x] true = false.

…
```

行列

- アイデア：行列は有限ドメインの関数として形式化する

'M[R]_(m, n)：環Rの要素を持つm×n行列の型
'M_n：n次正方行列
'rV_n：n行ベクトル

```
Variable R : Type. Variables m n : nat.
Inductive matrix : predArgType :=
      Matrix of {ffun 'I_m * 'I_n → R}.
```

{0,..., m-1}

{0,..., n-1}

- 'F_2による行ベクトルのハミング重みの求め方
  - 行ベクトルからその関数のグラフを取り出して，1を数える

```
Definition wH (v : 'rV_n) :=
     count F2_to_bool (fgraph [ffun x ⇒ (v 0) x]).
```

```
Definition distH (x y : 'rV_n) := wH (x - y).
```

置換行列

- 置換

'S_n : {0,.., n-1} のすべての置換集合

```
Inductive perm_type : predArgType :=
Perm (pval : {ffun T → T}) & injectiveb pval.
```

- 置換行列

perm_mx s : 置換sから得られるn x n置換行列

```
Lemma wH_perm_mx :
  ∀ n (s : 'S_n) z, wH (z *_m perm_mx s) = wH z.
Proof.
…
```

概要

- SSR の基礎
  - ssralgの階層
  - 基礎的なデータ構造
  - 例 : ハミング重み・距離
- 線形符号を定義してみよう
  - 線形代数から得られる性質
  - 最近傍復号についての簡単な定理
- McElieceを定義してみよう
  - ゴッパ符号を仮定・乱数生成は無視
- 結論

線形符号

- 定義してみよう

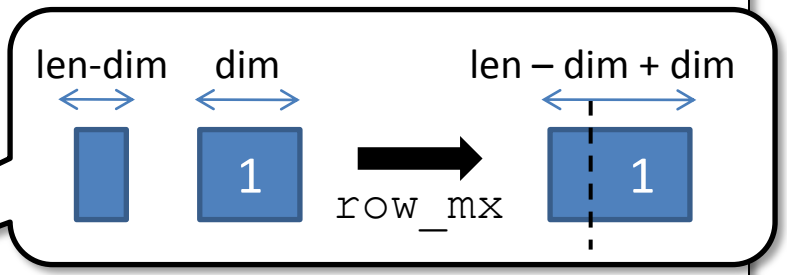
```
Record linearCode := LinearCode {
  dim : nat ;
  len' : nat ; len := len' + 1;
  Hdimlen : dim < len ;

  A : 'M['F_2]_(len - dim, dim) ;
  H : 'M_(len - dim, len) := cast_cols subnKC (ltnW (Hdimlen))
       (row_mx A (1%:M)) ;
  G : 'M_(dim, len) := cast_cols subnKC (ltnW Hdimlen)
       (row_mx 1%:M (-A)^T) ;

  C := [set s : 'rV_len | H *m s^T == 0] ;
  encode : 'rV_dim → 'rV_len := fun x ⇒ x *_m G ;

  d : nat ;
  d_min : ∀ i, i ∈ C → i ≠ 0 → d ≤ wH i ;
  d_min' : ∃ c, c ∈ C ∧ wH c = d
}.
```

次元と長さ



符号とエンコーダ

最小距離

線形符号の性質

- エンコーダは単射的
  - 補題(SSReflectを使って簡単に証明できる)

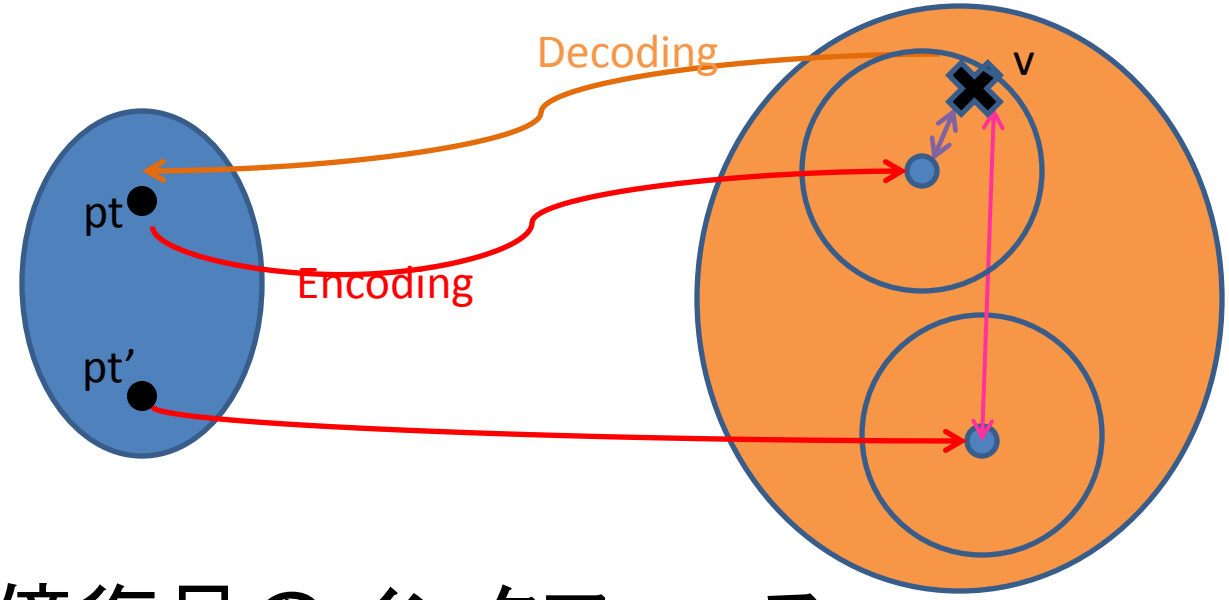
```
Lemma full_rank_inj : ∀ m n (A : 'M[F]_(m, n)),
 m < n → ¥rank A = m →
  ∀ (a b : 'rV[F]_m), a *_m A = b *_m A  →  a = b.
```

```
Lemma code_rank : ∀ c : linearCode,
      ¥rank (G c) = dim c.
```

- 定理

```
Variable c : linearCode.
Theorem encode_inj : ∀ a b,
       (encode c) a = (encode c) b → a = b.
Proof.
…
```

最近傍復号

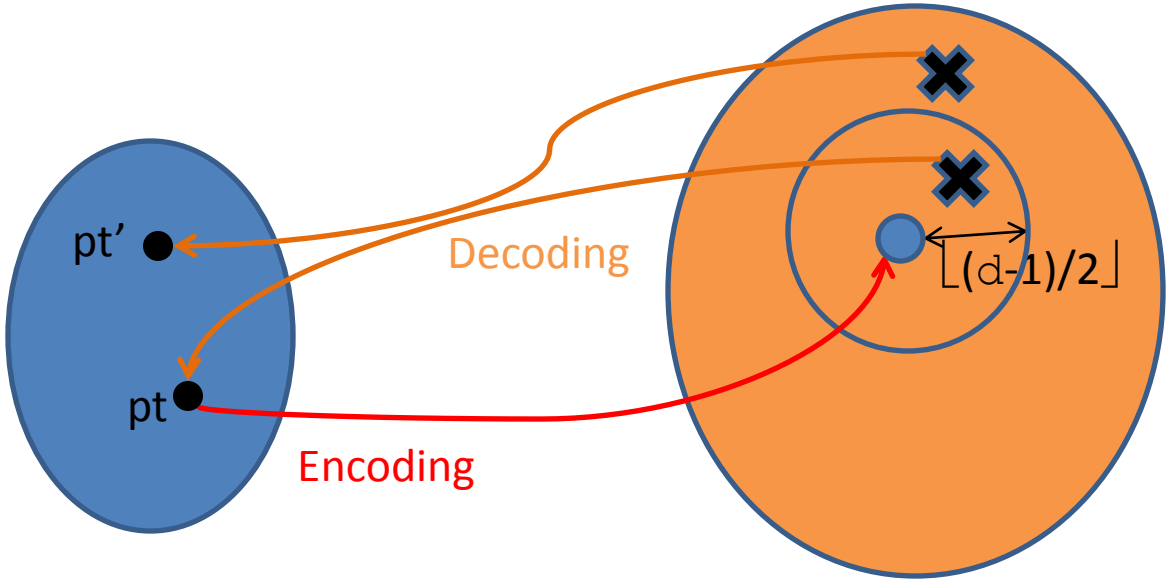


- 最近傍復号のインタフェース:

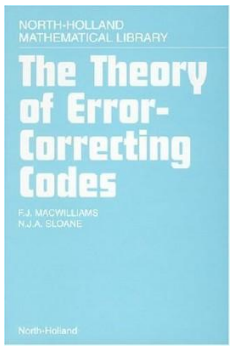
```
Record decoding (c : linearCode) := Decoding {
  decode : 'rV_(len c) → 'rV_(dim c) ;
  Hdecode_nearest : ∀ (v : 'rV_(len c)) pt,
    decode v = pt → ∀ pt',
    distH ((encode c) pt) v ≤ distH ((encode c) pt') v
}.
```

最近傍復号
簡単な定理

- 最小距離dの線形符号は⌊(d-1)/2⌋個の誤りを訂正できる



最近傍復号
簡単な定理



のp.10

- 訂正できる誤りの数

```
Variable c : linearCode.
Definition nnd_err_cor :=
  if odd (d c) then (d c).-1 %/ 2 else (d c).-2 %/ 2.
```

- 定理:

```
Variable decoder : decoding c.
Definition decode := decode _ decoder.
Lemma encode_decode' : ∀ pt v,
  distH ((encode c) pt) v ≤ nnd_err_cor c →
  decode v = pt.
```

```
Lemma encode_decode : ∀ pt v,
  wH v ≤ nnd_err_cor c →
  decode ((encode c) pt + v) = pt.
```

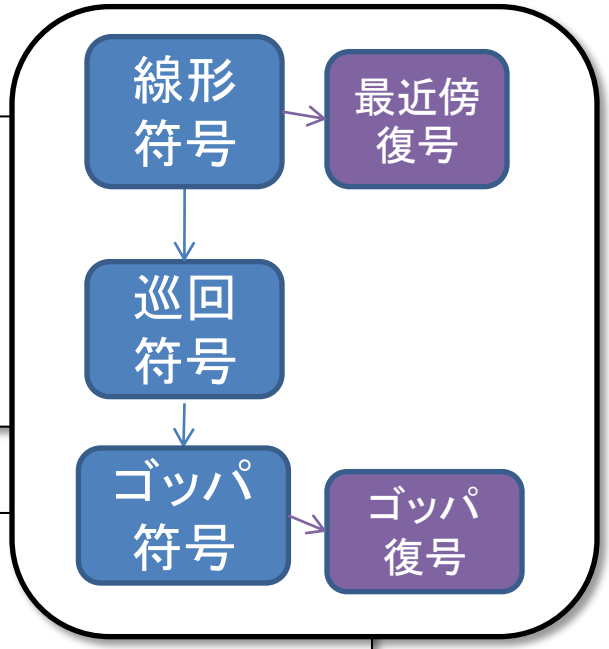
概要

- SSR の基礎
  - ssralgの階層
  - 基礎的なデータ構造
  - 例 : ハミング重み・距離
- 線形符号を定義してみよう
  - 線形代数から得られる性質
  - 最近傍復号についての簡単な定理
- McElieceを定義してみよう
  - ゴッパ符号を仮定・乱数生成は無視
- 結論

Goppa符号

```
Record cyclicCode := CyclicCode {
  c :> linearCode ;
  Hcyclic_code : ∀ (x : 'rV_(len c)),
    x ∈ C c ↔ right_cyclic_shift x ∈ C c
}.
```

```
Record goppaCode := GoppaCode {
  c :> cyclicCode
  (* and the property of being Goppa (in progress) *)
}.
```




```
Record decoding (g : goppaCode) := Decoding {
  decode : 'rV_(len g) -> 'rV_(dim g) ;
  gd_err_cor : nat ;
  Hgd_err_cor : 2 * gd_err_cor + 1 ≤ d g ;
  Hencode_decode : ∀ pt v, wH v ≤ gd_err_cor →
       decode ((encode g) pt + v) = pt
}.
```

McEliece暗号の鍵

```
Parameter c : Goppa_m.goppaCode.
Definition G := LinearCode_m.G c.
Definition n := LinearCode_m.len c.
Definition k := LinearCode_m.dim c.

Parameter gc_decoder : GoppaDecoding_m.decoding c.
Definition t := GoppaDecoding_m.gd_err_cor _ gc_decoder.
Definition decode := GoppaDecoding_m.decode _ gc_decoder.

Parameter S : 'M['F_2]_k.
Parameter S_inv : S ∈ unitmx.

Parameter p : 'S_n.
Definition P : 'M['F_2]_k := perm_mx p.

Definition G_hat : 'M_(k, n) := S *m G *m P.
```

Aさんは(n,k)線形符号Cを選ぶ（t個の誤りを訂正できる効率の良い復号アルゴリズムが存在する）

ランダムな正則行列S

ランダムな置換行列P

秘密鍵 = <S,G,P>

公開鍵 = <G_hat, t>

McElieceの暗号化

- Bさんの**平文**: kビットのベクトル

```
Parameter msg : 'rV['F_2]_k.
```

- Bさん はt個の1を持つnビットのランダムなエラーベクトルを選ぶ

```
Parameter z : 'rV['F_2]_n.
Parameter Hz : wH z = t.
```

- **暗号文**:

```
Definition cyp : 'rV_n := msg *_m G_hat + z.
```

McElieceの復号

- 復号:

```
Definition cyp_hat : 'rV_n := cyp *_m P^-1.

Definition msg_hat := decode cyp_hat.

Definition msg' : 'rV_k := msg_hat *_m invmx S.
```

- 暗号化・復号の正しさ:

```
Lemma decryption_undoes_encryption : msg = msg'.
Proof.
…
Qed.
```

概要

- SSR の基礎
  - ssralgの階層
  - 基礎的なデータ構造
  - 例 : ハミング重み・距離
- 線形符号を定義してみよう
  - 線形代数から得られる性質
  - 最近傍復号についての簡単な定理
- McElieceを定義してみよう
  - ゴッパ符号を仮定・乱数生成は無視
- 結論

結論

- SSReflectの代数ライブラリのおかげで, 従来定理証明支援器Coqで難しかった符号ベース暗号の形式化が楽になった
  - 今回話さなかった符号: 反復符号, ハミング符号
- 今後の課題:
  - 符号の階層の整備
    - グッパ符号の形式化
  - McElieceの安全性の証明の形式化