Mathematical Components の入門

集中講義 千葉大学大学院

アフェルト レナルド

産業技術総合研究所

2017 年 01 月 26 日

目的

- Mathematical Components[GMT08, GAA+13, Tas14, Mah14] による形式化を紹介する
- Mathematical Components のライブラリは:
  - 約 4 千の形式定義, 1 万 3 千の形式定理
  - 15 万行の Coq スクリプト
  - 奇数位数定理の 250 頁の証明は約 4 万行となる
  - 約 130 個のファイル

  $\Rightarrow$ 包括な説明は無理がある
- 群論のラグランジュ定理を用いて，Mathematical Components の基本的な使い方を説明する

アウトライン

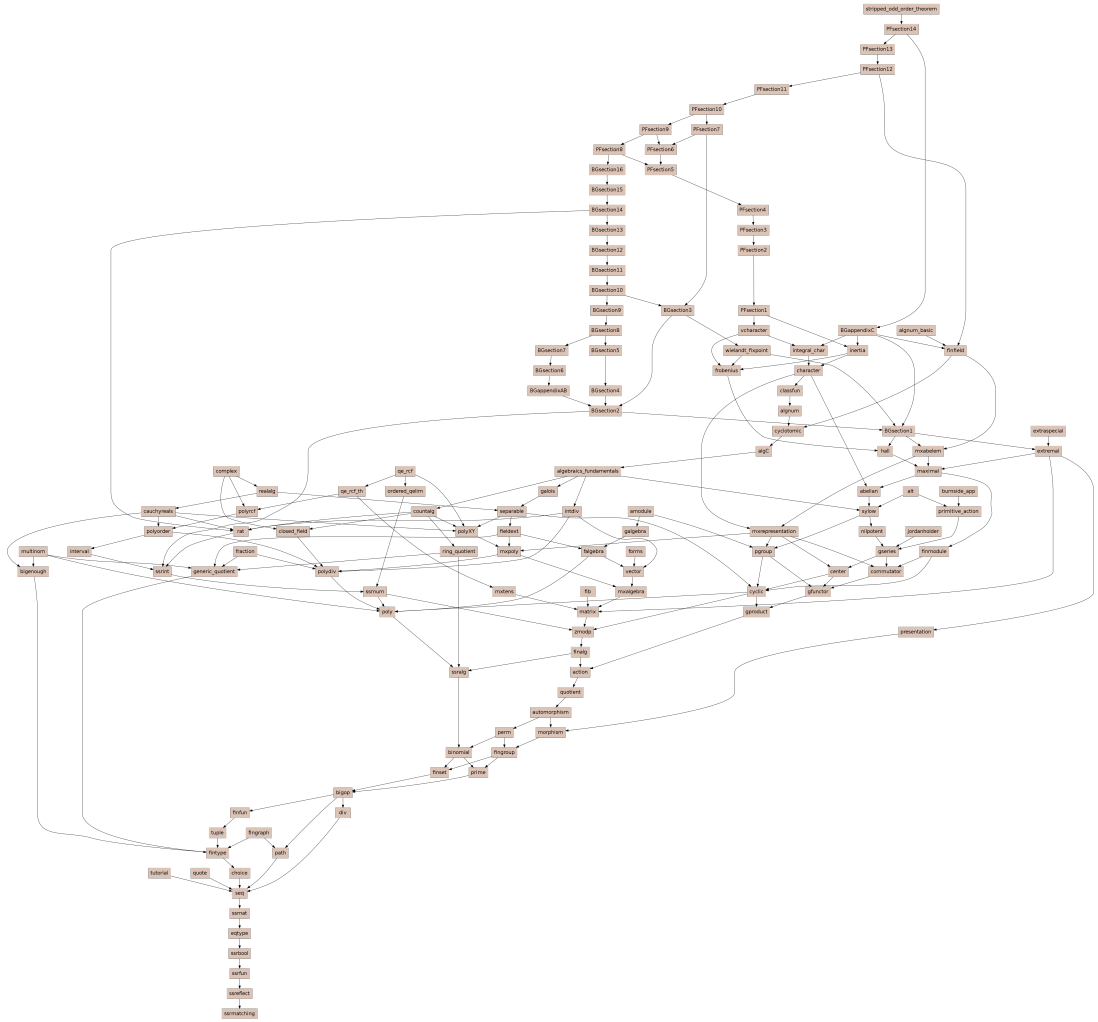
ssreflect.ml4: 新タクティク

Coq と比較

Coq	SSReflect
intro intros	move=>
revert	move:
generalize	move: (lem a)
specialize	move/(_ x)
rewrite	rewrite move=> ->
rewrite <-	rewrite - move=> <-
unfold	rewrite /
fold	rewrite -/
cutrewrite	rewrite (_ : a =b)
destruct injection	case
case_eq	case H :

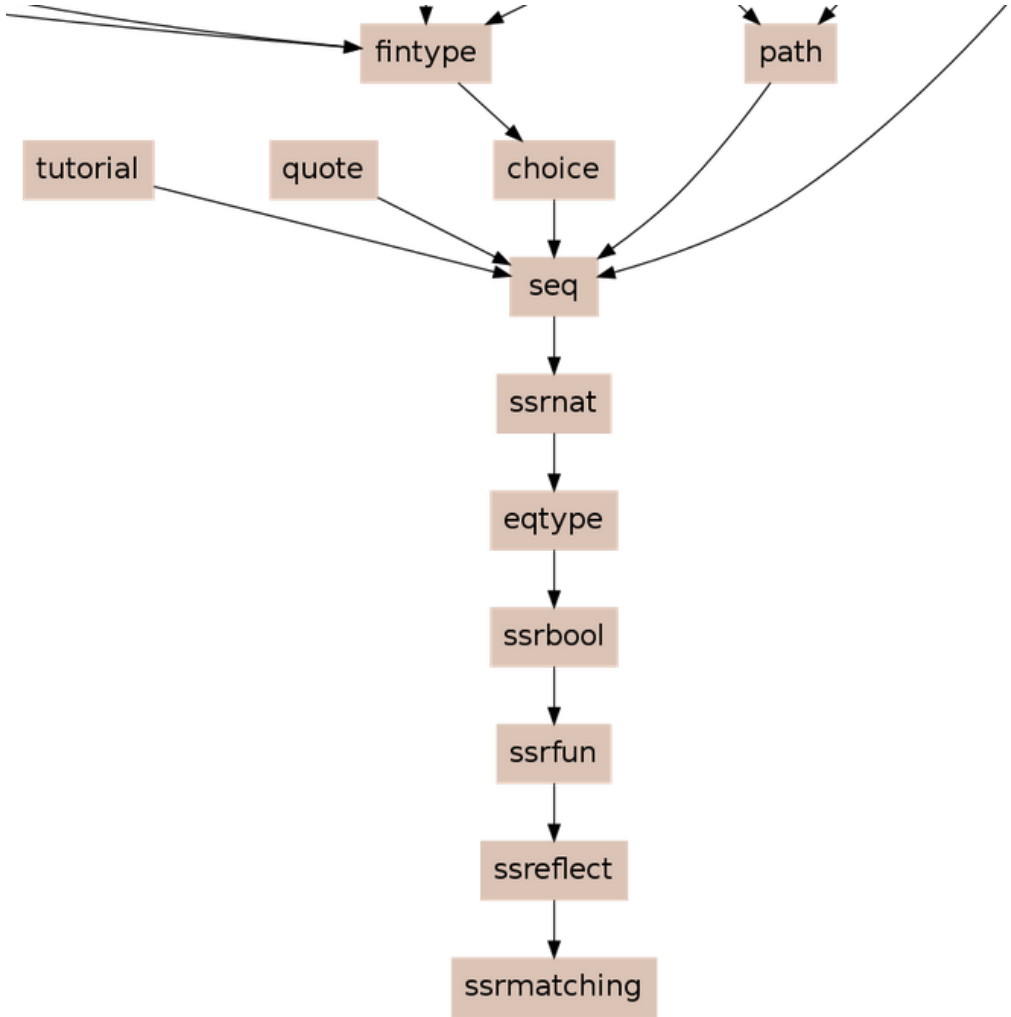
Coq	SSReflect
apply refine	apply:
exact	exact:
assert	have suff wlog
simpl	move=> /= rewrite /=
clear H	{H}
elim induction	elim
now	by done move=> // rewrite //
discriminate	
assumption	
contradiction	
pattern	rewrite [...]H set
f_equal	congr

Mathematical Components ライブラリ



図は `http://ssr.msr-inria.inria.fr/~jenkins/current/index.html` より

Mathematical Components の基礎ライブラリ



図は `http://ssr.msr-inria.inria.fr/~jenkins/current/index.html` より

ssrbool.v: リフレクションの実現

- is_true コアーションによって, bool 型は Prop 型に見える
- ビューの仕組み
- reflect 述語はブールの世界と Prop の世界の等価性を表す:

```
Inductive reflect (P : Prop) : bool -> Set :=
  | ReflectT : P -> reflect P true
  | ReflectF : ~ P -> reflect P false.
```

  - reflect 補題の例: andP, orP, negP, implyP

eqtype.v: 決定可能な同値関係 (1/2)

- 同値関係が決定可能なら, ブール値等式として定義ができる
  - 例えば, Coq の標準ライブラリの `Nat.eqb`
  - 一方, 実数の場合, 同値関係が決定可能ではない
- そのブール値等式と Leibniz 同値関係の等価性が証明できれば, その型は `eqType` として登録できる
  1. `reflect` 補題を利用
  2. 次の `Structure` のインスタスを作る

```
Module Equality.
Definition axiom T (e : rel T) :=
  forall x y, reflect (x = y) (e x y).
Structure mixin_of T :=
  Mixin {op : rel T; _ : axiom op}.
...
```

  3. canonical structure として宣言する

eqtype.v: 決定可能な同値関係 (2/2)

- canonical structure によって, 多重定義ができる [MT13]
  - 記法: eqType なら, ブール値等式は「==」と書ける
    - 「==」は Equality.mixin_of の op
  - 補題:
    - move/eqP で, Leibniz 同値関係とブール値等式を変換
    - ==の仮定 H の書き換え: rewrite (eqP H)

ssrfun.v, ssrnat.v, seq.v: 整理された基本ファイル

- ssrnat.v: 自然数のライブラリの再実装
  - 帰納的型より実行可能な関数 (例えば、leq)
  - Notation "n .+1":=(succn n)
  - 規則的な命名の基本
- seq.v: リストのライブラリの再実装
- ssrfun.v: 共通定義
  - 例えば:

```
Definition commutative op :=
  forall x y, op x y = op y x.
```

fintype.v: 有限な数の要素のある型

- 要素のリストを取り出す:
  - T : finType なら, T の要素のリストは enum T と書く
- finType だと, 抽象的なアルゴリズムは記述しやすくなる:
  - 全称記号: [forall x, P] (ビュー: forallP)
  - 存在記号: [exists x, P] (ビュー: existsP)
- eqType のように, 有限な数の要素のある型を finType として登録できる

```
Section RawMixin.
Variable T : eqType.
Definition axiom e :=
  forall x : T, count_mem x e = 1.
Record mixin_of := Mixin {
  mixin_base : Countable.mixin_of T;
  mixin_enum : seq T;
  _ : axiom mixin_enum
}.
```

アウトライン

有限集合の概要

- Mathematical Components の元の目的は有限群論であったため，有限集合のライブラリが充実している
- そのライブラリで構成できる有限集合の要素の型は `finType` 型に制限されている
- `T` 型を持つ要素の有限集合 S_0 と S_1 は次のように宣言できる:

```
Variable T : finType.
Variables S0 S1 : {set T}.
```

- 一言で説明すると，`T` は `finType` 型であるので，`{set T}`は指示関数の型として考えれば良い

有限集合のライブラリ

有限集合を構成:

- 空集合は `set0` と書く
- 共通部分 $S_0 \cap S_1$ は `S0 :&: S1` と書く
- 写像 f による集合 S_0 の像 (つまり，$f(S_0)$) は `f @: S0` と書く
- 直積集合 $S_0 \times S_1$ は `setX S0 S1` を書く

有限集合の基本的な性質:

- S_0 の濃度は `#|S0|` と書く
- 帰属関係 $a \in S_0$ は `a \in S0` と書く
- 真部分集合 $S_0 \subsetneq S_1$ は `S0 \proper S1` と書く
- 部分集合 $S_0 \subseteq S_1$ は `S0 \subset S1` と書く

アウトライン

有限群の概要

有限群は次のように宣言する:

```
Variable gT : finGroupType.
Variable G : {group gT}.
```

- gT は群の元の型となる (global finite container として考える)
- finGroupType に属する型は群の基本的な性質の一部を持つ
  - fingroup.v, Module FinGroup
- 有限群そのものは{group gT}という型を持つ
  - {group gT}は{set gT}のサブタイプ (fingroup.v, 1319 行目)
  - ```
    Definition group_set A :=
      (1 \in A) && (A * A \subset A).
    ```

    (fingroup.v, 1129 行目)
    ```

群の定義に当たる基本的な要素

- 二項演算は関数 mulg またはそれに当たる*記法となる．例えば，g と h が G の元とする:

```
Variables g h : gT.
Hypotheses
  (gG : g \in G) (hG : h \in G).
```

そうすると，g * h は型 gT を持つ Check 命令で確認できる:

```
Check g * h : gT.
```

g * h は G の元であることは補題 groupM から得る:

```
Check groupM gG hG : g * h \in G.
```

- 二項演算の結合法則は次の補題になる:

```
Lemma mulgA : associative mulg.
```

- 単位元は oneg gT またはそれに当たる 1%g 記法となる．逆元は invg g またはそれに当たる g^-1 記法となる．

有限群に関する補題例

補題名	言明 (G : {group gT})
mul1g	1 * x = x
mulg1	x * 1 = x
mulgA	x * (y * z) = x * y * z
mulgV	x * x^-1 = 1
mulVg	x^-1 * x = 1
invgK	x^-1^-1 = x
invMg	(x * y)^-1 = y^-1 * x^-1
group1	1 \in G
groupV	(x^-1 \in G) = (x \in G)
groupM	x \in G -> y \in G -> x * y \in G

部分群, 剰余類

- 上記の G の部分群 H は次のように宣言する:

```
Variable H : {group gT}.
Hypothesis HG : H \subset G.
```

- G を有限群とし，H を部分集合とする．このとき，$g \in G$ に対する左剰余類 $gH = \{g \star h \,|\, h \in H\}$ に当たる形式定義は `lcoset H g` または `g *: H` と書く
- 左剰余類の集合 $(G/H)_l$ は `lcosets H G` と書く

ラグランジュ定理

- 教科書通りの言明:
  - 「G は有限群とし，H を G の部分群とする．このとき，H の濃度は，G の濃度を割り切る.」
  - この商を部分群の指数といい，$[G : H]$ と書く．つまり，$|G| = [G : H] \times |H|$ が成り立つ.
- Mathematical Components での証明:
  - $[G : H]$ は H の剰余類の数として定義できる．H の剰余類は G を分割し，それぞれの剰余類は H と同じ濃度があるため，ラグランジュ定理が成り立つ.
- 今回の証明:
  - 次の補題を示す: 「G を有限群とし，H と K を $K \subsetneq H$ を満たす G の部分群とするとき，$[G : K] = [G : H] \times [H : K]$.」
  - 最後に, $\mathtt{K}$ は $\{1\}$ という部分集合すればよい.

アウトライン

左剰余類の共通の元はないこと

言明

次の補題を証明し, 有限集合と有限群のライブラリを使ってみる

- L0 と L1 を 2 つの左剰余類とする. その際, L0 と L1 の共通の元があれば, L0=L1 が成り立つ:

```
Lemma coset_disjoint L0 L1 :
  L0 \in lcosets H G ->
  L1 \in lcosets H G ->
  L0 :&: L1 != set0 -> L0 = L1.
```

左剰余類の共通の元はないこと

証明の開始

- SSReflect のタクティクを用いて, 証明を開始する:

```
move/lcosetsP.
case.
move=> g0.
move=> g0G.
move=> TMP.
rewrite TMP.
rewrite {TMP}.
rewrite {L0}.
```

- 一行でまとめる:

```
case/lcosetsP => g0 g0G ->{L0}.
```

アウトライン

終域 $(G/H)_l$ の単射 (1/2)

- 次の単射を定義したい:

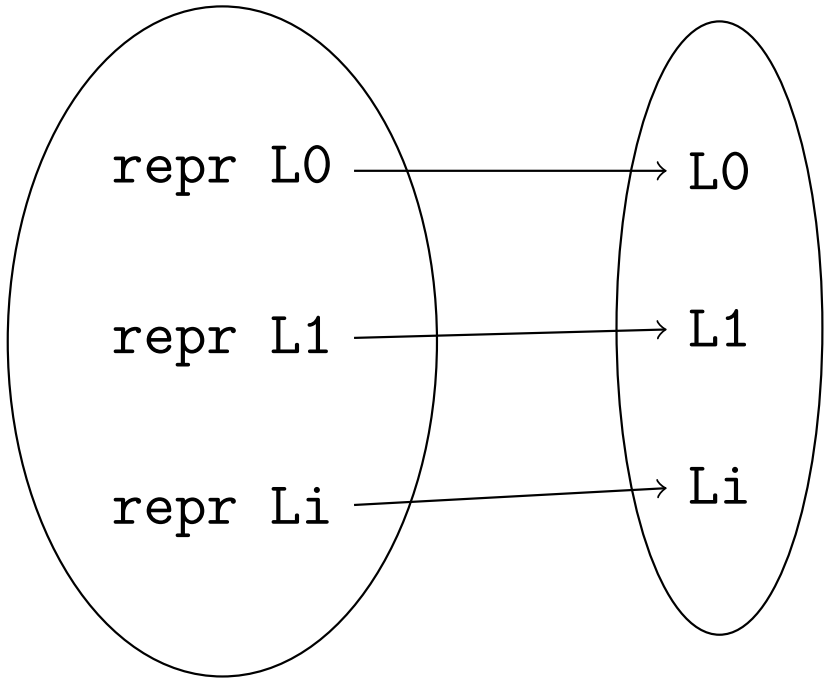
$$\begin{aligned} \alpha &: \mathcal{G} \to (G/H)_l \\ &: g \mapsto gH \end{aligned}$$

- そのため, $\mathcal{G}$ を次のように定義する:

```
Definition reprs := repr @: lcosets H G.
```

  - `repr X` は集合 `X` の代表的な要素を返す.

終域 $(G/H)_l$ の単射 (2/2)



reprs
(紙上の記法: $\mathcal{G}$)

lcosets H G
(紙上の記法: $(G/H)_l$)

終域 $(G/H)_l$ の単射の性質

α は全単射であることを証明する:

- injection:

```
Lemma injective_coset : {in reprs &,
  injective (fun x => x *: H)}.
```

  - `{in ... &, injective ...}`記法を確認する
  - `imsetP` 補題: $y \in f(D) \leftrightarrow \exists x, x \in D \wedge y = f(x)$

- surjection:

```
Lemma surj :
  (fun x => x *: H) @: reprs = lcosets H G.
```

アウトライン

部分群の指数の推移関係の言明

- 紙上の言明:

  G を有限群とし，H と K を $K \subsetneq H$ を満たす G の部分群とするとき，$[G : K] = [G : H] \times [H : K]$.

- 形式的な言明:

```
Variable gT : finGroupType.
Variables G H K : {group gT}.
Hypotheses (HG : H \subset G)
  (KG : K \subset G) (HK : K \proper H).

Lemma index_trans :
  #| G : K | = #| G : H | * #| H : K |.
```

部分群の指数の推移関係の証明

- 次の単射を用意する:
  - $\alpha \quad : \quad \mathcal{G} \to (G/H)_l$
  - $\quad : \quad g \mapsto gH$
  - $\beta \quad : \quad \mathcal{H} \to (H/K)_l$
  - $\quad : \quad h \mapsto hK$
  - $\phi \quad : \quad \mathcal{G} \times \mathcal{H} \to (G/K)_l$
  - $\quad : \quad (g, h) \mapsto (g \star h)K$
- 上記の α, β, ϕ は全射なので,
  $|(G/K)_l| = |\mathcal{G} \times \mathcal{H}| = |\mathcal{G}| \times |\mathcal{H}| = |(G/H)_l| \times |(H/K)_l|$
- 次のステップを確かめる: 「Si $ghK = g'h'K$ avec $g, g' \in \mathcal{G}$ et $h, h' \in \mathcal{H}$, on en déduit $g'^{-1}ghK = h'K \ldots$」

アウトライン

ラグランジュ定理の形式証明

```
Variable gT : finGroupType.
Variables G H : {group gT}.
Hypothesis HG : H \subset G.

Theorem Lagrange :
  #| G | = #| H | * #| G : H |.
```

アウトライン

結論

- Mathematical Components の他の教材:
  - [Aff15] (特に, 有限群に関する補足)
  - [MT16] (夏に発表された)
- Mathematical Components の他の応用例:
  - 情報理論 [AHS14]
  - 符号理論 [AG15, AGS16]
  - 幾何学 (線型代数学)[AC17]

参考文献 I

Reynald Affeldt and Cyril Cohen, *Formal foundations of 3d geometry to model robot manipulators*, Proceedings of the6th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2017), Paris, France, January 16–17, 2017, ACM Press, Jan. 2017, pp. 30–42.

Reynald Affeldt, 定理証明支援系 *Coq* による形式検証, Available at: `https://staff.aist.go.jp/reynald.affeldt/ssrcoq/`, Jul. 2015, 集中講義. 京都大学大学院理学研究科数学・数理解析専攻数理解析系.

Reynald Affeldt and Jacques Garrigue, *Formalization of error-correcting codes: from hamming to modern coding theory*, Proceedings of the 6th International Conference on Interactive Theorem Proving, ITP 2015, Nanjing, China, August 24–27, 2015, Lecture Notes in Computer Science, vol. 9236, Springer, 2015, pp. 17–33.

Reynald Affeldt, Jacques Garrigue, and Takafumi Saikawa, *Formalization of reed-solomon codes and progress report on formalization of ldpc codes*, Proceedings of theInternational Symposium on Information Theory and Its Applications (ISITA 2016), Monterey, California, USA, October 30–November 2, 2016, IECE, Oct. 2016, pp. 537–541.

Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues, *Formalization of Shannon's theorems*, J. Autom. Reasoning **53** (2014), no. 1, 63–103.

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry, *A machine-checked proof of the odd order theorem*, Proceedings of the 4th International Conference on Interactive Theorem Proving, ITP 2013, Rennes, France, July 22–26, 2013, Lecture Notes in Computer Science, vol. 7998, Springer, 2013, pp. 163–179.

Georges Gonthier, Assia Mahboubi, and Enrico Tassi, *A small scale reflection extension for the Coq system*, Tech. Report RR-6455, INRIA, 2008, Version 14 (March 2014).

参考文献 II


Assia Mahboubi, *Computer-checked mathematics: a formal proof of the odd order theorem*, Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, Vienna, Austria, July 14–18, 2014, ACM, Jul. 2014, Article no. 4.


Assia Mahboubi and Enrico Tassi, *Canonical structures for the working Coq user*, Proceedings of the 4th International Conference on Interactive Theorem Proving, ITP 2013, Rennes, France, July 22–26, 2013, Lecture Notes in Computer Science, vol. 7998, Springer, 2013, pp. 19–34.


———, *Mathematical components*, 2016, Available at: `https://math-comp.github.io/mcb/book.pdf`.


Enrico Tassi, *Mathematical components, a large library of formalized mathematics*, Workshop on Formalization of Mathematics in Proof Assistants, Institut Henri Poincaré, May 2014, Oral presentation.