# Ax-Grothendieck in lean

Joseph Hua

June 11, 2022

Imperial College London

# Motivation

# Polynomial maps

**Definition (Polynomial maps)**

Polynomial maps on a field $K$ are regular endomorphisms on $K^n$, i.e. $n$ polynomials in $K[x_1, \ldots, x_n]$.

# Polynomial maps

**Definition (Polynomial maps)**

Polynomial maps on a field $K$ are regular endomorphisms on $K^n$, i.e. $n$ polynomials in $K[x_1, \ldots, x_n]$.

**Examples**

- Surjective but not injective: $f : \mathbb{C} \to \mathbb{C} := x \mapsto x^2$

# Polynomial maps

**Definition (Polynomial maps)**

Polynomial maps on a field $K$ are regular endomorphisms on $K^n$, i.e. $n$ polynomials in $K[x_1, \ldots, x_n]$.

**Examples**

- Surjective but not injective: $f : \mathbb{C} \to \mathbb{C} := x \mapsto x^2$
- Neither surjective nor injective: $f : \mathbb{C}^2 \to \mathbb{C}^2 := (x, y) \mapsto (x, xy)$

# Polynomial maps

**Definition (Polynomial maps)**

Polynomial maps on a field $K$ are regular endomorphisms on $K^n$, i.e. $n$ polynomials in $K[x_1, \dots, x_n]$.

**Examples**

- Surjective but not injective: $f : \mathbb{C} \to \mathbb{C} := x \mapsto x^2$
- Neither surjective nor injective: $f : \mathbb{C}^2 \to \mathbb{C}^2 := (x, y) \mapsto (x, xy)$
- Bijective: $f : \mathbb{C}^3 \to \mathbb{C}^3 := (x, y, z) \mapsto (x, y, z + xy)$

# Polynomial maps

## Definition (Polynomial maps)

Polynomial maps on a field $K$ are regular endomorphisms on $K^n$, i.e. $n$ polynomials in $K[x_1, \ldots, x_n]$.

## Examples

- Surjective but not injective: $f : \mathbb{C} \to \mathbb{C} := x \mapsto x^2$
- Neither surjective nor injective: $f : \mathbb{C}^2 \to \mathbb{C}^2 := (x, y) \mapsto (x, xy)$
- Bijective: $f : \mathbb{C}^3 \to \mathbb{C}^3 := (x, y, z) \mapsto (x, y, z + xy)$

## Code

```
def poly_map (K : Type*) [comm_semiring K] (n : ℕ) : Type* :=
fin n → mv_polynomial (fin n) K

def eval : poly_map K n → (fin n → K) → (fin n → K) :=
λ ps as k, mv_polynomial.eval as (ps k)
```

# Ax-Grothendieck

**Theorem (Ax-Grothendieck)**
*Any injective polynomial map over an algebraically closed field is surjective. In particular injective polynomial maps over $\mathbb{C}$ are surjective.*

**Theorem (Ax-Grothendieck)**
*Any injective polynomial map over an algebraically closed field is surjective. In particular injective polynomial maps over $\mathbb{C}$ are surjective.*

```
theorem Ax_Groth {n : ℕ} {ps : poly_map K n}
  (hinj : function.injective (poly_map.eval ps)) :
function.surjective (poly_map.eval ps) := sorry
```

# Ax-Grothendieck

**Theorem (Ax-Grothendieck)**
*Any injective polynomial map over an algebraically closed field is surjective. In particular injective polynomial maps over $\mathbb{C}$ are surjective.*

```
theorem Ax_Groth {n : ℕ} {ps : poly_map K n}
  (hinj : function.injective (poly_map.eval ps)) :
function.surjective (poly_map.eval ps) := sorry
```

- Amazing fact: the Lefschetz principle implies that if we show this for a single algebraically closed field of characteristic $n$ (a model of $\mathsf{ACF}_n$) and it will be true for any model of $\mathsf{ACF}_n$ ($n$ is zero or prime).

# Ax-Grothendieck

**Theorem (Ax-Grothendieck)**
*Any injective polynomial map over an algebraically closed field is surjective. In particular injective polynomial maps over $\mathbb{C}$ are surjective.*

```
theorem Ax_Groth {n : ℕ} {ps : poly_map K n}
  (hinj : function.injective (poly_map.eval ps)) :
function.surjective (poly_map.eval ps) := sorry
```

- Amazing fact: the Lefschetz principle implies that if we show this for a single algebraically closed field of characteristic $n$ (a model of $\mathsf{ACF}_n$) and it will be true for any model of $\mathsf{ACF}_n$ ($n$ is zero or prime).
- Amazing fact: Lefschetz also says that if we show it for all $\mathsf{ACF}_p$ for large $p$ then it is also true for $\mathsf{ACF}_0$.

# Ax-Grothendieck

**Theorem (Ax-Grothendieck)**
*Any injective polynomial map over an algebraically closed field is surjective. In particular injective polynomial maps over $\mathbb{C}$ are surjective.*

```
theorem Ax_Groth {n : ℕ} {ps : poly_map K n}
  (hinj : function.injective (poly_map.eval ps)) :
function.surjective (poly_map.eval ps) := sorry
```

- Amazing fact: the Lefschetz principle implies that if we show this for a single algebraically closed field of characteristic $n$ (a model of $\mathsf{ACF}_n$) and it will be true for any model of $\mathsf{ACF}_n$ ($n$ is zero or prime).
- Amazing fact: Lefschetz also says that if we show it for all $\mathsf{ACF}_p$ for large $p$ then it is also true for $\mathsf{ACF}_0$.
- Good news: We can easily show it for algebraic closures of $\mathbb{F}_p$ for any $p$.

# Locally finite fields

**Definition (Locally finite fields)**

Let $K$ be a field of characteristic $p$ a prime. Then the following are equivalent definitions for $K$ being a *locally finite field*:

1. The minimal subfield generated by any finite subset of $K$ is finite.

# Locally finite fields

**Definition (Locally finite fields)**
Let $K$ be a field of characteristic $p$ a prime. Then the following are equivalent definitions for $K$ being a *locally finite field*:

1. The minimal subfield generated by any finite subset of $K$ is finite.
2. $\mathbb{F}_p \to K$ is algebraic.

# Locally finite fields

**Definition (Locally finite fields)**
Let $K$ be a field of characteristic $p$ a prime. Then the following are equivalent definitions for $K$ being a *locally finite field*:

1. The minimal subfield generated by any finite subset of $K$ is finite.
2. $\mathbb{F}_p \to K$ is algebraic.
3. $K$ embeds into an algebraic closure of $\mathbb{F}_p$.

# Locally finite fields

**Definition (Locally finite fields)**

Let $K$ be a field of characteristic $p$ a prime. Then the following are equivalent definitions for $K$ being a *locally finite field*:

1. The minimal subfield generated by any finite subset of $K$ is finite.
2. $\mathbb{F}_p \to K$ is algebraic.
3. $K$ embeds into an algebraic closure of $\mathbb{F}_p$.

# Locally finite fields

**Definition (Locally finite fields)**
Let $K$ be a field of characteristic $p$ a prime. Then the following are equivalent definitions for $K$ being a *locally finite field*:

1. The minimal subfield generated by any finite subset of $K$ is finite.
2. $\mathbb{F}_p \to K$ is algebraic.
3. $K$ embeds into an algebraic closure of $\mathbb{F}_p$.

**Theorem**
*Locally finite fields satisfy Ax-Grothendieck.*

**Proof.**

$$\mathbb{F}_p \hookrightarrow \mathbb{F}_p(\texttt{coeffs}) \hookrightarrow K$$

$$\downarrow_{\mathbb{F}_p(\texttt{coeffs})} \text{ respects injectivity}$$

$\square$

**Theorem (Lefschetz principle)**

*Let $\phi$ be a sentence in the language of rings. Then the following are equivalent:*

1. *Some model of* $\mathsf{ACF}_0$ *satisfies* $\phi$*. (If you like* $\mathbb{C} \vDash \phi$*.)*
2. $\mathsf{ACF}_0 \vDash \phi$
3. *There exists* $n \in \mathbb{N}$ *such that for any prime* $p$ *greater than* $n$*,* $\mathsf{ACF}_p \vDash \phi$
4. *There exists* $n \in \mathbb{N}$ *such that for any prime* $p$ *greater than* $n$*, some model of* $\mathsf{ACF}_p$ *satisfies* $\phi$*.*

# Model Theory

# Languages

```
structure Language : Type (u+1) :=
  (functions : ℕ → Type u)
  (relations : ℕ → Type u)
```

# Languages

```
structure Language : Type (u+1) :=
  (functions : ℕ → Type u)
  (relations : ℕ → Type u)
```

```
inductive ring_consts : Type*
| zero : ring_consts
| one : ring_consts

inductive ring_unaries : Type*
| neg : ring_unaries

inductive ring_binaries : Type*
| add : ring_binaries
| mul : ring_binaries

def ring_funcs : ℕ → Type*
| 0 := ring_consts
| 1 := ring_unaries
| 2 := ring_binaries
| (n + 3) := pempty

def ring_signature : Language :=
(Language.mk) (ring_funcs)
  (λ n, pempty)
```

# Languages

```
structure Language : Type (u+1) :=
  (functions : ℕ → Type u)
  (relations : ℕ → Type u)
```

## More examples

- Lanugage of groups
- Language of monoid actions from a monoid $M$ and modules on a ring $A$
- Single binary relations

```
inductive ring_consts : Type*
| zero : ring_consts
| one : ring_consts

inductive ring_unaries : Type*
| neg : ring_unaries

inductive ring_binaries : Type*
| add : ring_binaries
| mul : ring_binaries

def ring_funcs : ℕ → Type*
| 0 := ring_consts
| 1 := ring_unaries
| 2 := ring_binaries
| (n + 3) := pempty

def ring_signature : Language :=
(Language.mk) (ring_funcs)
  (λ n, pempty)
```

# Terms and formulas

```
inductive bounded_preterm (n : ℕ) : ℕ → Type u
  | x_ : ∀ (k : fin n), bounded_preterm 0
  | bd_func : ∀ {l : ℕ} (f : L.functions l), bounded_preterm l
  | bd_app : ∀ {l : ℕ} (t : bounded_preterm (l + 1))
      (s : bounded_preterm 0), bounded_preterm l
```

```
def bounded_term (n : ℕ) := bounded_preterm L n 0
```

$x_1 * 0 \rightsquigarrow$ `bd_app (bd_app (bd_func mul) (x_ 1)) (bd_func zero)`

# Proof overview

An overview of the proof:

$$\begin{tikzcd}
\text{(algebraic) } \chi_p \text{ Ax-G} \arrow[d, "\text{soundness}"] & & \\
\text{(model th.) } \chi_p \text{ Ax-G} \arrow[d, "\mathsf{ACF}_p \text{ is complete}"] & & \\
\text{(model th.) } \chi_p \text{ Ax-G} \arrow[r, "\text{completeness}"'] \arrow[d, "\text{Lefschetz } \chi\text{-change}"] & \text{(algebraic) } \chi_p \text{ Ax-G} \arrow[dr] & \\
\text{(model th.) } \chi_0 \text{ Ax-G} \arrow[r, "\text{completeness}"'] & \text{(algebraic) } \chi_0 \text{ Ax-G} \arrow[r, "\text{case on } \chi"] & \text{(algebraic) Ax-G}
\end{tikzcd}$$