# A Small Reflection On

## Group Automorphisms

François Garillot

Mathematical Components

Microsoft Research - INRIA Joint Centre
Orsay, France

# The Coq Proof Assistant

# The COQ Proof Assistant + SSReflect

V 1.1 : Just released !

`http://www.msr-inria.inria.fr/Projects/math-components`

# The Coq Proof Assistant + SSReflect

V 1.1 : Just released !

`http://www.msr-inria.inria.fr/Projects/math-components`

- A renewed tactic shell *: faster proofs (to write)*
  - better bookkeeping
  - better (hierarchical) layout
  - ”surgical” rewriting

# The Coq Proof Assistant + SSReflect

V 1.1 : Just released !

`http://www.msr-inria.inria.fr/Projects/math-components`

- A renewed tactic shell *: faster proofs (to write)*
  - better bookkeeping
  - better (hierarchical) layout
  - ”surgical” rewriting
- Small Scale Reflection
  - We often work on a *decidable* domain, where e.g. the excluded middle makes sense.
  - Coerce booleans to propositions.
  - Reflection : booleans $\leftrightarrow$ logical propositions

# The Coq Proof Assistant + SSReflect

V 1.1 : Just released !

`http://www.msr-inria.inria.fr/Projects/math-components`

- A renewed tactic shell *: faster proofs (to write)*
  - better bookkeeping
  - better (hierarchical) layout
  - ”surgical” rewriting
- Small Scale Reflection
  - We often work on a *decidable* domain, where e.g. the excluded middle makes sense.
  - Coerce booleans to propositions.
  - Reflection : booleans $\leftrightarrow$ logical propositions
- Libraries for dealing with equality, finite types, naturals, lists

# Formalising Some Finite Group Theory

# Formalising Some Finite Group Theory

- A project of Mathematical Components, with members at Sophia-Antipolis and Orsay (INRIA), Cambridge (MSR)
- Towards a formalisation of the Feit-Thompson (*'Odd Order'*) theorem

# Formalising Some Finite Group Theory

- A project of Mathematical Components, with members at Sophia-Antipolis and Orsay (INRIA), Cambridge (MSR)
- Towards a formalisation of the Feit-Thompson (*’Odd Order’*) theorem
- We already have a number of elements:
  - functions of finite sets
  - Groups and basic lemmata
  - Lagrange, isomorphism theorems
  - Sylow theorems
  - Frobenius Lemma
  - Schur-Zassenhaus theorem
  - Simplicity of the alternating group

# Formalising Some Finite Group Theory

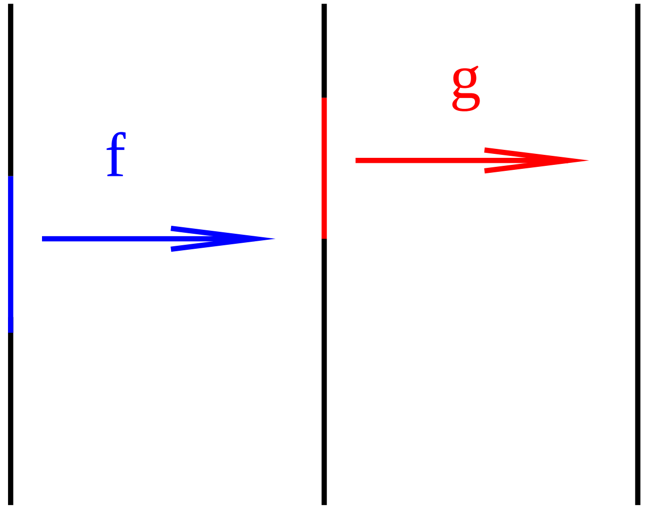
- A project of Mathematical Components, with members at Sophia-Antipolis and Orsay (INRIA), Cambridge (MSR)
- Towards a formalisation of the Feit-Thompson (*'Odd Order'*) theorem
- We already have a number of elements:
  - functions of finite sets
  - Groups and basic lemmata
  - Lagrange, isomorphism theorems
  - Sylow theorems
  - Frobenius Lemma
  - Schur-Zassenhaus theorem
  - Simplicity of the alternating group
- And counting ...

# Group Morphisms : mathematically

# Group Morphisms : mathematically

> *We call* morphism *from E to E' a mapping from E to E' s.t.* $f(x \bullet_E y) = f(x) \bullet_{E'} f(y)$ *for all* $x, y$ *in* $E \times E$*. The identity mapping is a morphism, the composition of two morphisms is a morphism* [Bourbaki]

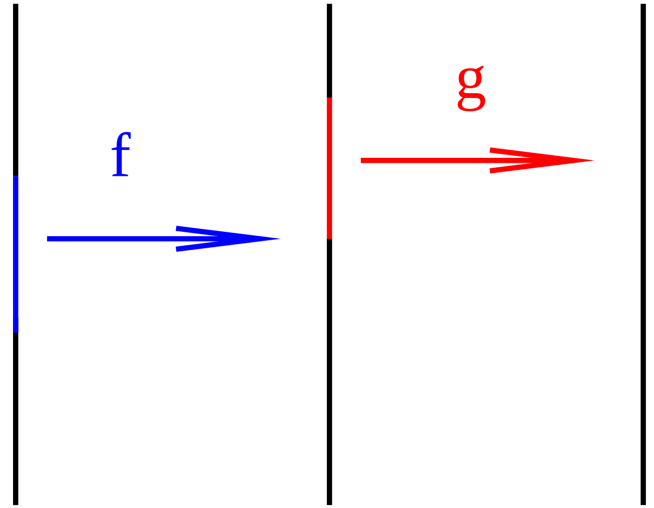
The *morphism* has a smaller *domain* than the underlying *function*.



# Group Morphisms : mathematically

> *We call* morphism *from E to E' a mapping from E to E' s.t.* $f(x \bullet_E y) = f(x) \bullet_{E'} f(y)$ *for all* $x, y$ *in* $E \times E$*. The identity mapping is a morphism, the composition of two morphisms is a morphism* [Bourbaki]

The *morphism* has a smaller *domain* than the underlying *function*.




`Canonical Structure qualid.`

`...`

> *Each time an equation of the form* $(x_i \ \_) =_{\beta\delta\iota\zeta} c_i$ *has to be solved during the type-checking process,* `qualid` *is used as a solution.* [Coq manual]

# Group Theory : Groups

```
Structure finGroupType : Type := FinGroupType {
  element :> finType;
  1 : element;
  _^-1 : element → element;
  _ • _ : element → element → element;
  unitP : ∀ x, 1 • x = x;
  invP : ∀ x, x^-1 • x = unit;
  mulP : ∀ x1 x2 x3, x1 • (x2 • x3) = (x1 • x2) • x3
}.
```

Two-staged development : a *carrier type* providing structural properties,

# Group Theory : Groups

```
Structure finGroupType : Type := FinGroupType {
  element :> finType;
  1 : element;
  _^-1 : element → element;
  _ • _ : element → element → element;
  unitP : ∀ x, 1 • x = x;
  invP : ∀ x, x^-1 • x = unit;
  mulP : ∀ x1 x2 x3, x1 • (x2 • x3) = (x1 • x2) • x3
}.
```

Two-staged development : a *carrier type* providing structural properties, and a *set* corresponding to the actual object

```
Variable elt : finGroupType.

Structure group : Type := Group {
  SoG :> setType elt;
  SoGP : 1 ⊂ SoG && (SoG :•: SoG) ⊂ SoG
}.
```

# Group Theory : Programming with Canonical Structures

`group_setI` $\simeq$ for all H, K groups, H $\cap$ K has the required group properties.

```
Canonical Structure setI_group := Group group_setI.
```

```
Coq < Check (_ ∩ _).
∩
     : forall T : finType, setType T → setType T → setType T
```

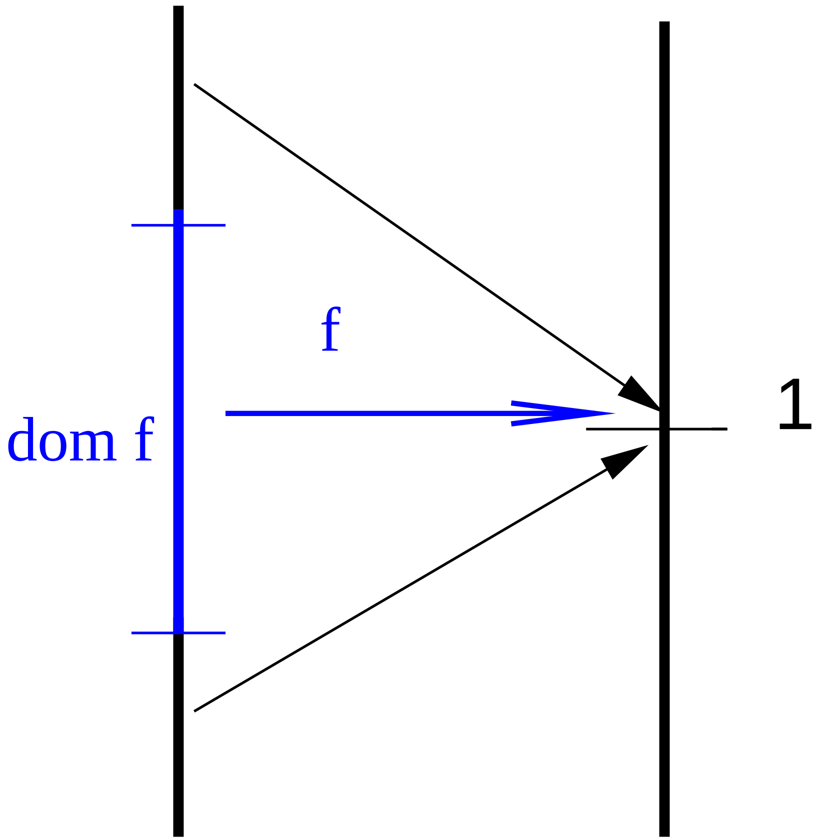
```
Lemma groupMl : ∀ (H:group _) x y, x ∈ H ⇒
(x • y) ∈ H = y ∈ H.
```

```
Lemma setI_stable : ∀ (H K : group _) x y, x ∈ (H ∩
  K) ⇒
  y ∈ (H ∩ K) ⇒ (x • y) ∈ (H ∩ K : setType _).
Proof. by move ⇒ x y Hx Hy; rewrite groupMl. Qed.
```

# Group Morphisms : in Coq

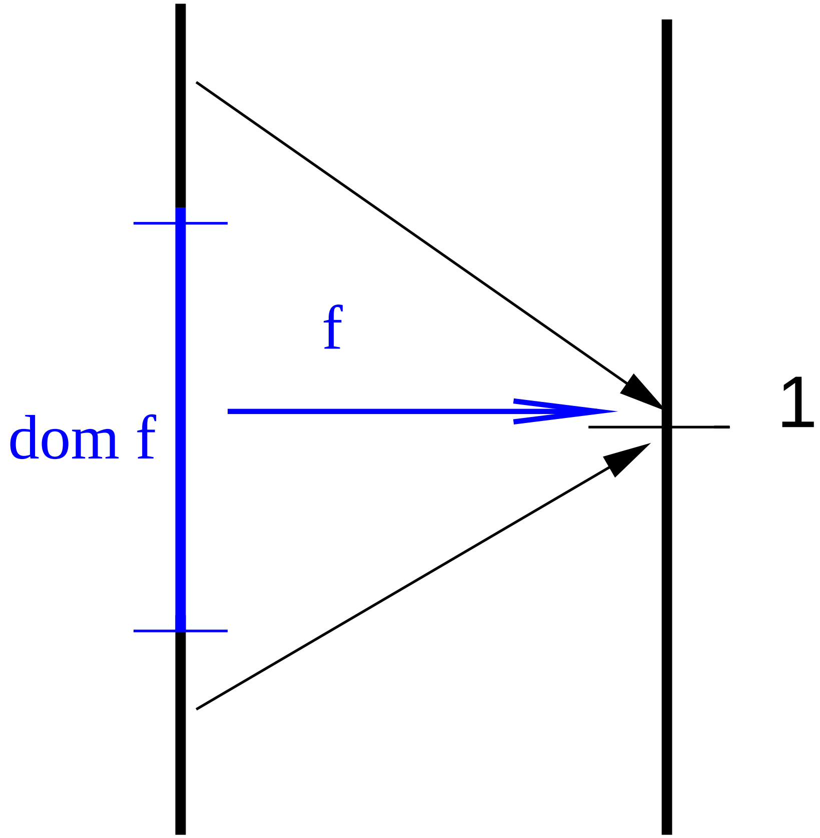
# Group Morphisms : in Coq

```
Definition ker f :=
{x| ∀y, f(x • y) == f(y) }
Definition dom f :=
ker f∪{x| f x != 1}.
```



# Group Morphisms : in Coq

```
Definition ker f :=
{x| ∀y, f(x • y) == f(y) }
Definition dom f :=
ker f∪{x| f x != 1}.
```




```
Structure morphism : Type := Morphism {
  mfun :> elt1 → elt2;
  group_set_dom : group_set (dom mfun);
  morphM :  morphic (dom f) mfun
}.
```

```
Definition morphic H f := ∀ x y,
    f x ∈ H →
    f y ∈ H →
    f (x • y) = f x * f y.
```

`morphic` $\simeq$ product commutation

# Group Morphism : discussion

- an unambiguous, dynamically built domain

# Group Morphism : discussion

- an unambiguous, dynamically built domain except for $x \mapsto 1$.

# Group Morphism : discussion

- an unambiguous, dynamically built domain except for $x \mapsto 1$.
- but pretty hard to create morphisms ex nihilo,

# Group Morphism : discussion

- an unambiguous, dynamically built domain except for $x \mapsto 1$.
- but pretty hard to create morphisms ex nihilo,
- proving properties on (canonical) *morphisms* is easy, but how do we export them to *morphic* functions ?

# Automorphism : Definition

[ Recall : a *morphism* is a *morphic* mapping on a *group*, and sends to the unit elsewhere. ]

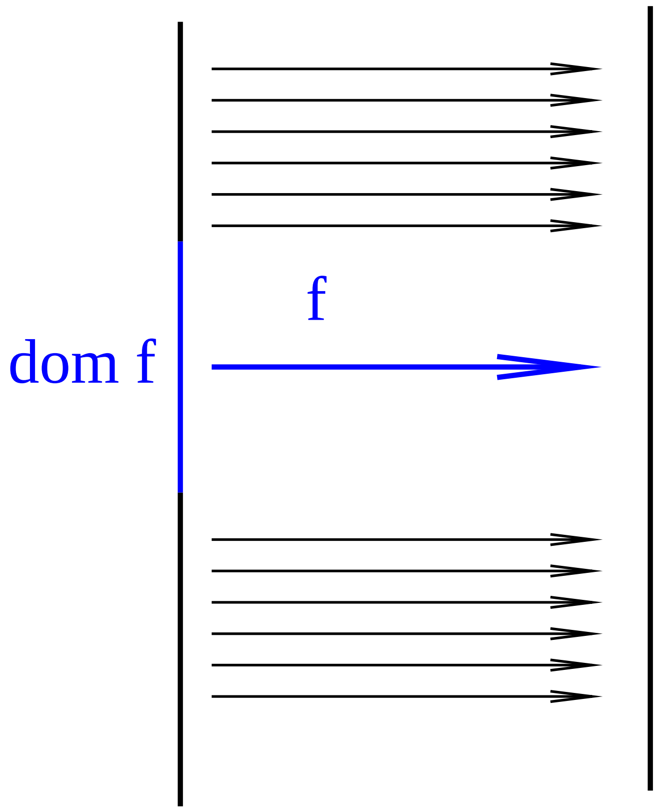
An *automorphism* is a *bijective endomorphism*.

We build them on bijective *functions* : by itself, they are *morphic*, but not *morphisms*.

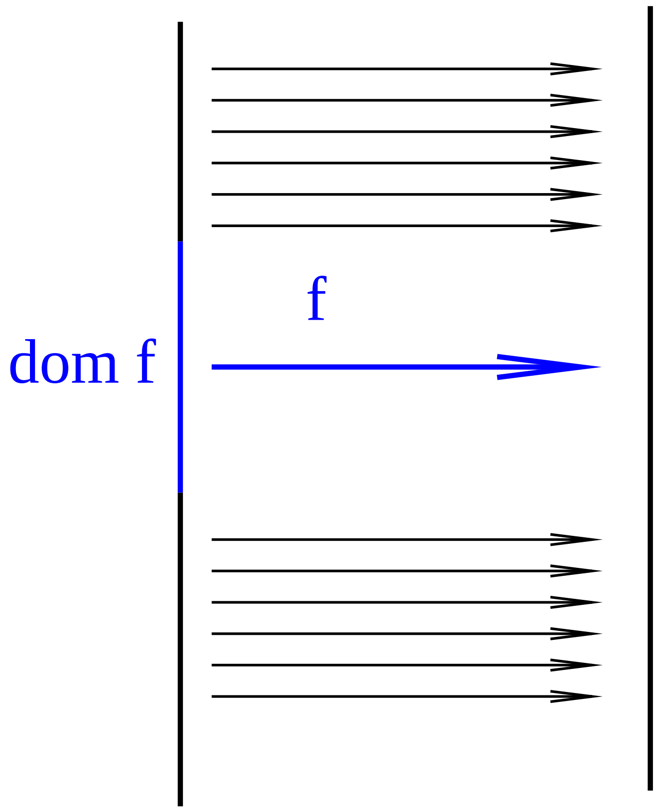
Automorphisms are defined as :

- *permutations* of a *group carrier type*
- morphic on a given subgroup
- coincide with the identity elsewhere (not the trivial morphism)

# Automorphisms : the need for a restriction

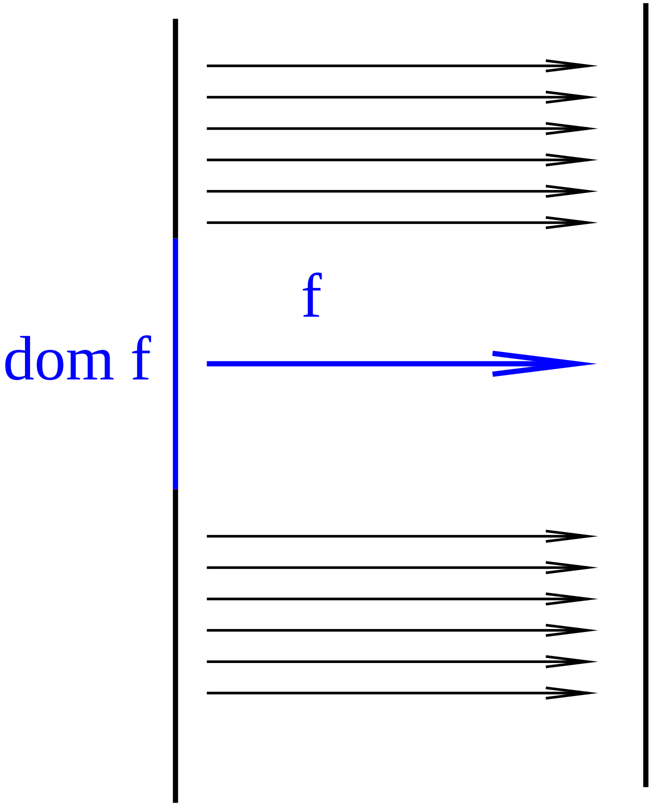


# Automorphisms : the need for a restriction



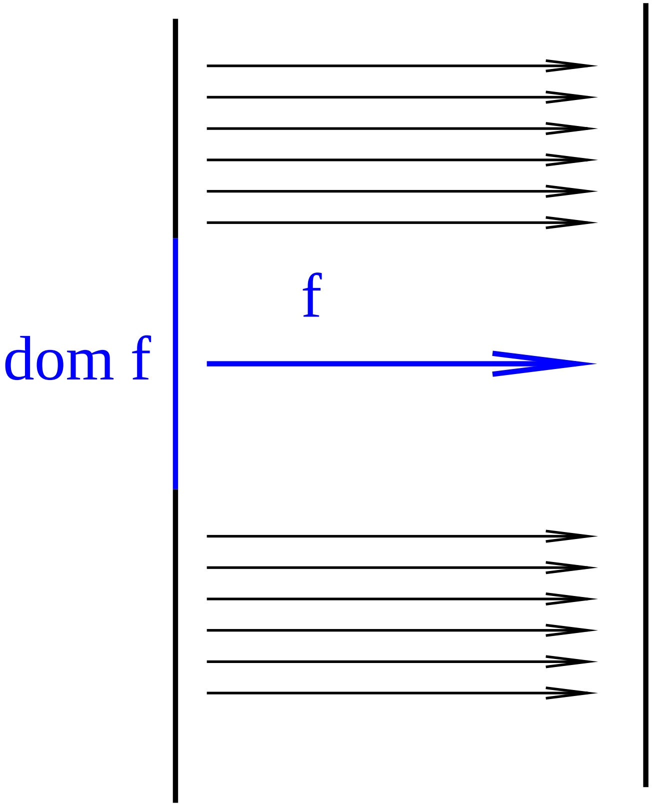

- our formalisation of automorphisms turns out to be very similar to what is done with morphisms,

# Automorphisms : the need for a restriction



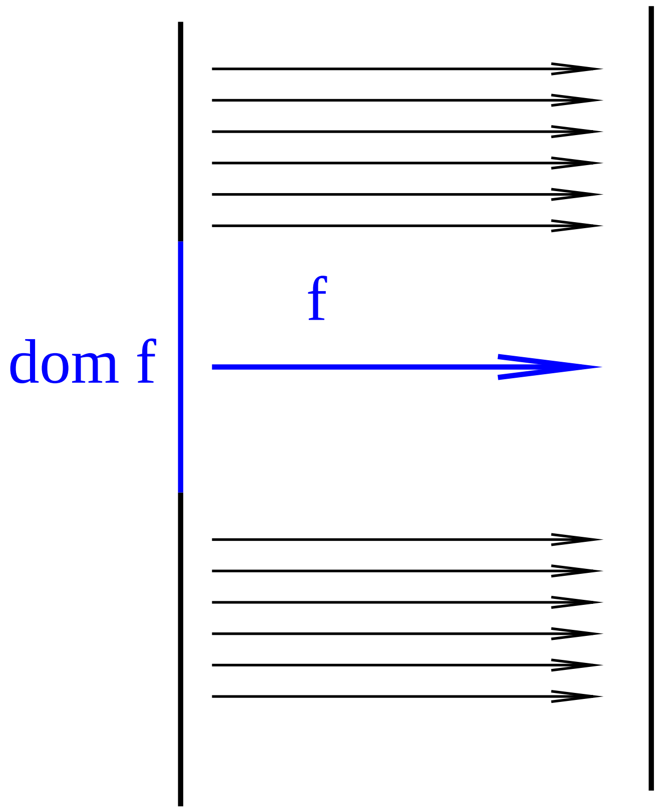

- our formalisation of automorphisms turns out to be very similar to what is done with morphisms,
- they could enjoy symmetric `ker` and `dom` notions,

# Automorphisms : the need for a restriction




- our formalisation of automorphisms turns out to be very similar to what is done with morphisms,
- they could enjoy symmetric `ker` and `dom` notions,
- coincide with a morphism on their ’domain’

# Automorphisms : the need for a restriction




- our formalisation of automorphisms turns out to be very similar to what is done with morphisms,
- they could enjoy symmetric `ker` and `dom` notions,
- coincide with a morphism on their 'domain'
- restrict morphic functions, obtain morphisms.

# Restriction of morphic functions : Default Values

# Restriction of morphic functions : Default Values

```
Definition mrestr f H :=
  [fun x ⇒ if (H x) then (f x) else 1].

Definition morphicrestr f H :=
  if ~~(morphic H f)
  then (fun _ ⇒1)
  else (mrestr f H).
```

# Restriction of morphic functions : Default Values

```
Definition mrestr f H :=
  [fun x ⇒ if (H x) then (f x) else 1].

Definition morphicrestr f H :=
  if ~~(morphic H f)
  then (fun _ ⇒1)
  else (mrestr f H).
```

```
Lemma morph1 :
∀ (f:morphism _ _), f 1 = 1.

Lemma dfequal_morphicrestr :
∀ x, x ∈ H ⇒
  (f x) = (morphicrestr f H).

Lemma morphic1 :
∀ (f: _ → _) (H: group _)
  (Hmorph: morphic H f),
  f 1 = 1.
Proof.
rewrite (dfequal_morphicrestr Hmorph);
[exact: morph1|exact:group1].
Qed.
```

# Morphism Restrictions : Discussion

- we have successfully adapted a number of results from morphisms (an internalised representation, defined with a `Canonical Structure`) to morphic functions (declarative expression of a local property of a function).
- but we have to treat the trivial case separately, sometimes extensively,
- however, this is usually simpler.
- scales up : automorphisms are permutations that behave well on a domain, and coïncide wih the identity elsewhere.

# Cyclic Groups : Application

# Cyclic Groups : Application

- A cyclic group is a *monogenous* group: the intersection of all groups containing a given singleton.
- Given rise to by the iterated multiplication of an element by itself:

$$C_p(a) = \{1, a, a \bullet a, a^3, \ldots, a^{(p-1)}\}$$

# Cyclic Groups : Application

- A cyclic group is a *monogenous* group: the intersection of all groups containing a given singleton.
- Given rise to by the iterated multiplication of an element by itself:

$$C_p(a) = \{1, a, a \bullet a, a^3, \ldots, a^{(p-1)}\}$$

- Automorphism group has nice properties, isomorphic to $\mathbb{Z}_\times^n$

# Cyclic Groups : Application

- A cyclic group is a *monogenous* group: the intersection of all groups containing a given singleton.
- Given rise to by the iterated multiplication of an element by itself:

$$C_p(a) = \{1, a, a \bullet a, a^3, \ldots, a^{(p-1)}\}$$

- Automorphism group has nice properties, isomorphic to $\mathbb{Z}^n_\times$
- $C_p(a)$ is isomorphic to $\mathbb{Z}/p\mathbb{Z}$

# Cyclic Groups : Application

- A cyclic group is a *monogenous* group: the intersection of all groups containing a given singleton.
- Given rise to by the iterated multiplication of an element by itself:

$$C_p(a) = \{1, a, a \bullet a, a^3, \ldots, a^{(p-1)}\}$$

- Automorphism group has nice properties, isomorphic to $\mathbb{Z}^n_\times$
- $C_p(a)$ is isomorphic to $\mathbb{Z}/p\mathbb{Z}$
- Hand-proven lemmas for cyclic groups in all their generality,

# Cyclic Groups : Application

- A cyclic group is a *monogenous* group: the intersection of all groups containing a given singleton.
- Given rise to by the iterated multiplication of an element by itself:

$$C_p(a) = \{1, a, a \bullet a, a^3, \ldots, a^{(p-1)}\}$$

- Automorphism group has nice properties, isomorphic to $\mathbb{Z}^n_\times$
- $C_p(a)$ is isomorphic to $\mathbb{Z}/p\mathbb{Z}$
- Hand-proven lemmas for cyclic groups in all their generality,
- . . . which usually are easier to prove on $\mathbb{Z}/p\mathbb{Z}$

# Conclusion

- a dialogue between proofs on a canonical notion of morphism and a localised property,
- real benefits obtained from `Canonical Structures`, here and elsewhere (Message to the Coq team : we want more !)
- a confirmation of the pertinence of our structures over a large development,
- interested about a more architectural way of doing this,