

A formalization of a simple example of a cofinitary group

Bart Kastermans

Department of Mathematics
University of Colorado–Boulder

bart.kastermans@colorado.edu
<http://www.bartk.nl/>

August 2009: BLAST

Outline

- ▶ Definitions and Basics on MCG
- ▶ Formalization
- ▶ The example
- ▶ The formalization

Definitions

S_∞ : the group of bijections $\mathbb{N} \rightarrow \mathbb{N}$ (permutations) with operation composition.

$f \in S_\infty$ is *cofinitary* iff f is the identity or has only finitely many fixed points.

$G \leq S_\infty$ is a *cofinitary group* iff all $g \in G$ are cofinitary.

$G \leq S_\infty$ is a *maximal cofinitary group* iff G is a cofinitary group and is not properly contained in another cofinitary group.

Some basic properties

Any cofinitary group is contained in a maximal cofinitary group.

(Adeleke, Truss) A maximal cofinitary group cannot be countable.

(Neumann) There is a cofinitary group of size $|\mathbb{R}|$.

(Zhang) If $|\mathbb{N}| < \kappa \leq |\mathbb{R}|$ then it is consistent that there is a maximal cofinitary group G with $|G| = \kappa$.

Some more results (Kastermans)

No maximal cofinitary group has infinitely many orbits.

MA implies there is a maximal cofinitary group with multiple infinite orbits.

MA implies there is a locally finite maximal cofinitary group.

The axiom of constructibility implies there exists a coanalytic maximal cofinitary group.

What is formalization?

Want to construct actual proofs of results like the ones mentioned above.

Were these results then accepted without proof?

What is a proof?

System chosen

Isabelle/HOL

Isabelle is a generic proof assistant.

HOL is for higher order logic.

Chosen because of Isar language.

Proof General

The image shows a desktop environment with two windows. The background window is a Terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Help). The terminal text shows the user navigating to the Desktop/Isabelle directory and starting Isabelle with Emacs. The foreground window is Emacs, titled "emacs: Scratch.thy", with a menu bar (File, Edit, View, Cmds, Tools, Options, Buffers, Proof-General, Isabelle, X-Symbol, Help). The Emacs interface includes a toolbar with icons for State, Content, Retract, Undo, Next, Use, Goto, Find, Command, Stop, Restart, and Info. The main editing area shows the text of a proof script with an error message. The error message is as follows:

```
Raw--*-XEmacs: Scratch.thy (Isar script XS:isar/s Font;)--Top-----
(1) (initialization/error) An error has occurred while loading /home/kasterma/.
emacs:

Symbol's function definition is void: scroll-bar-mode

To ensure normal operation, you should investigate the cause of the error
in your initialization file and remove it. Use the '-debug-init' option
to XEmacs to view a complete error backtrace.
```

At the bottom of the Emacs window, the status bar displays: Binary--*-XEmacs: *Warnings* (Fundamental;)--All-----

```
emacs: Demo.thy
File Edit View Cmds Tools Options Buffers Proof-General X-Symbol Isabelle
State Context Retract Undo Next Use Goto Find Command Stop Restart Info
catheory.thy Demo.thy
theory Demo
imports Main
begin

lemma ex1: "A & B -> B & A";
proof (rule impI)
  assume "A & B"
  from this show "B & A"
  proof (rule conjE)
    assume a: "A" and b: "B"
    from b and a show "B & A" by (rule conjI);
  qed;
qed;

prf ex1;

Raw-----XEmacs: Demo.thy (Isar script XS:isar/s Font; Scr
(((thm.HOL.impI . _) . _) .
  (A(H: _).
    (((conjE . _) . _) . _) . H) .
    (A{(H: _) (Ha: _)}).
      (((conjI . _) . _) . Ha) . H))))))
```

```
lemma ex2: "A ∧ B → B ∧ A"
proof
  assume "A ∧ B"
  from this show "B ∧ A"
  proof
    assume "A" and "B"
    thus ?thesis by auto;
  qed;
qed;
```

```
lemma ex3: "A ∧ B → B ∧ A" by auto;
```

```
U
lemma ex4: assumes "A ∧ B"
  shows "B ∧ A"
proof
  from `A ∧ B` show "B" ..;
  from `A ∧ B` show "A" ..;
qed;
```

```
lemma assumes Pf: " $\exists x. P(f\ x)$ " shows " $\exists y. P\ y$ "
proof -
  from Pf obtain x where "P(f x)" ..
  thus " $\exists y. P\ y$ " ..
qed;
```

```
lemma intcom2: "A ∩ B = B ∩ A" by fast;
```

The Simple Example

Let $g(k) = k + 1$ on the integers. Let $E_{X_1} = \langle g \rangle$. Pick a bijection from the natural numbers to the integers (say evens map to the positive integers, and the odds to the negative integers). Then conjugating $E_{X_1}1$ by that bijection gives a cofinitary group E_{X_2} .

```
definition S_inf :: "(nat  $\Rightarrow$  nat) set"
where
  "S_inf = {f::(nat  $\Rightarrow$  nat). bij f}"; (* next;; *)
```

```
locale CofinitaryGroup =
  fixes
    dom :: "(nat  $\Rightarrow$  nat) set"
  assumes
    type_dom : "dom  $\subseteq$  S_inf" and
    id_com : "id  $\in$  dom" and
    mult_closed : "f  $\in$  dom  $\wedge$  g  $\in$  dom  $\implies$  f  $\circ$  g  $\in$  dom" and
    inv_closed : "f  $\in$  dom  $\implies$  inv f  $\in$  dom" and
    cofinitary : "f  $\in$  dom  $\wedge$  f  $\neq$  id  $\implies$  finite (Fix f)"; (* ne
```

```
definition upOne :: "int  $\Rightarrow$  int"  
where  
  "upOne n = n + 1";  (* next;; *)
```

```
theorem bij_upOne: "bij upOne"  
by (unfold bij_def, rule conjI [OF inj_upOne surj_upOne]);
```

```
theorem "Fix upOne = {}"  
proof -  
  from Fix_def[of upOne]  
  have "Fix upOne = {n . upOne n = n}" by auto;  
  with no_fix_upOne have "Fix upOne = {n . False}" by auto;  
  with Set.empty_def show "Fix upOne = {}" by auto;  
qed;
```

```
inductive_set Ex1 :: "(int  $\Rightarrow$  int) set" where
base_func: "upOne  $\in$  Ex1" |
comp_func: "f  $\in$  Ex1  $\Rightarrow$  (upOne  $\circ$  f)  $\in$  Ex1" |
comp_inv : "f  $\in$  Ex1  $\Rightarrow$  ((inv upOne)  $\circ$  f)  $\in$  Ex1";
```

```
theorem Ex1_Normal_form: "(f ∈ Ex1) = (∃k. ∀n. f(n) = n + k)";
proof
  assume "f ∈ Ex1"
  with Ex1_Normal_form_part1 [of f]
    show "(∃k. ∀n. f(n) = n + k)" by auto;
next;
  assume "∃k. ∀n. f(n) = n + k"
  with Ex1_Normal_form_part2
    show "f ∈ Ex1" by auto;
qed;    (* next;; *)
```

```
theorem no_fixed_pt:
  assumes f_Ex1: "f ∈ Ex1"
  and f_not_id: "f ≠ id"
  shows "Fix f = {}";      (*;
```

```

theorem closed_comp: "f ∈ Ex1 ∧ g ∈ Ex1 ⇒ f ∘ g ∈ Ex1" (* ne
proof (rule Ex1.induct [of f], blast);
  assume "f ∈ Ex1 ∧ g ∈ Ex1";
  with Ex1.comp_func[of g] show "upOne ∘ g ∈ Ex1" by auto;
next
  fix fa
  assume "fa ∘ g ∈ Ex1"
  with Ex1.comp_func [of "fa ∘ g"]
    and Fun.o_assoc [of "upOne" "fa" "g"]
    show "upOne ∘ fa ∘ g ∈ Ex1" by auto;
next
  fix fa
  assume "fa ∘ g ∈ Ex1"
  with Ex1.comp_inv [of "fa ∘ g"]
    and Fun.o_assoc [of "inv upOne" "fa" "g"]
    show "(inv upOne) ∘ fa ∘ g ∈ Ex1" by auto;
qed;

```

```
definition ni_bij:: "nat  $\Rightarrow$  int"  
where  
"ni_bij n = (if ((n mod 2) = 0)  
              then int (n div 2)  
              else -int (n div 2) - 1)"
```

```
theorem ni_bij_bij: "bij ni_bij"; (* next *)
proof (unfold bij_def, rule conjI);

  show INJ: "inj ni_bij"
  proof (rule injI)
    fix x::nat and y::nat
    assume eq_ass: "ni_bij x = ni_bij y";
    show "x = y"
```

```

theorem conj_fix_pt: " $\forall f::('a \Rightarrow 'b). \forall g::('b \Rightarrow 'b). (bij\ f) \Rightarrow ((inv\ f)\ (Fix\ g)) = Fix\ ((inv\ f)\ \circ\ g\ \circ\ f)$ "; ■(* next; *)
proof -
  fix f::"'a  $\Rightarrow$  'b"
  assume bij_f: "bij f"
  with bij_def have inj_f: "inj f" by auto;
  fix g::"'b  $\Rightarrow$  'b"
  show " $((inv\ f)\ (Fix\ g)) = Fix\ ((inv\ f)\ \circ\ g\ \circ\ f)$ ";
  thm set_eq_subset[of " $(inv\ f)\ (Fix\ g)$ " " $Fix((inv\ f)\ \circ\ g\ \circ\ f)$ "]
  proof
    show " $(inv\ f)\ (Fix\ g) \subseteq Fix\ ((inv\ f)\ \circ\ g\ \circ\ f)$ "
    proof
      fix x
      assume "x  $\in$   $(inv\ f)\ (Fix\ g)$ "
      with image_def have " $\exists y \in Fix\ g. x = (inv\ f)\ y$ " by auto;

```

```
definition CONJ :: "(int ⇒ int) ⇒ (nat ⇒ nat)"
where
"CONJ f = (inv ni_bij) ∘ f ∘ ni_bij";  (* next;; *)
declare CONJ_def [simp] -- "automated tools can use the definition";
```

```

lemma type_CONJ: "f ∈ Ex1 ⇒ (inv ni_bij) ∘ f ∘ ni_bij ∈ S_inf"
(* next;; *)
proof -
  assume f_Ex1: "f ∈ Ex1"
  with all_bij have "bij f" by auto;
  with ni_bij_bij and comp_bij
    have bij_f_nibij: "bij (f ∘ ni_bij)" by auto;
  with ni_bij_bij and bij_imp_bij_inv have "bij (inv ni_bij)" by auto;
  with bij_f_nibij and comp_bij[of "f ∘ ni_bij" "inv ni_bij"]
    and o_assoc[of "inv ni_bij" "f" "ni_bij"]
    have "bij ((inv ni_bij) ∘ f ∘ ni_bij)" by auto;
  with S_inf_def show "((inv ni_bij) ∘ f ∘ ni_bij) ∈ S_inf"; by auto;
qed;

```

```

lemma inv_CONJ:
  assumes bij_f: "bij f"
  shows "inv (CONJ f) = CONJ (inv f)" (is "?left = ?right")
(* next; *)
proof -
  have st1: "?left = inv ((inv ni_bij) ◦ f ◦ ni_bij)"
    using CONJ_def by auto;
  from ni_bij_bij and bij_imp_bij_inv
  have inv_ni_bij_bij: "bij (inv ni_bij)" by auto;
  with bij_f and comp_bij have "bij (inv ni_bij ◦ f)" by auto;
  with o_inv_distrib[of "inv ni_bij ◦ f" ni_bij] and ni_bij_bij
  have "inv ((inv ni_bij) ◦ f ◦ ni_bij) =
    (inv ni_bij) ◦ (inv ((inv ni_bij) ◦ f))" by auto;
  with st1 have st2: "?left =
    (inv ni_bij) ◦ (inv ((inv ni_bij) ◦ f))" by auto;
  from inv_ni_bij_bij and `bij f` and o_inv_distrib
  have h1: "inv (inv ni_bij ◦ f) = inv f ◦ inv (inv (ni_bij))" by auto;
  from ni_bij_bij and inv_inv_eq[of ni_bij]
  have "inv (inv ni_bij) = ni_bij" by auto;
  with st2 and h1 have "?left = (inv ni_bij ◦ (inv f ◦ (ni_bij)))" by auto;
  with o_assoc have "?left = inv ni_bij ◦ inv f ◦ ni_bij" by auto;
  with CONJ_def[of "inv f"] show ?thesis by auto;
qed;

```

```
definition Ex2 :: "(nat  $\Rightarrow$  nat) set"
where
  "Ex2 = CONJ`Ex1";
(* next: *)
```

```

theorem Ex2_cofinitary:
  assumes f_Ex2: "f ∈ Ex2"
  and f_nid: "f ≠ id"
  shows "Fix f = {}";
(* next; *)
proof -
  from f_Ex2 and mem_Ex2_rule
  obtain g where g_Ex1: "g ∈ Ex1" a
  with id_CONJ and f_nid have "g ≠
  with g_Ex1 and no_fixed_pt[of g]
  from conj_fix_pt[of ni_bij g] and
  have "(inv ni bij) ` (Fix g) = Fix f

```

```

lemma comp_Ex2:
  assumes f_Ex2: "f ∈ Ex2" and
  g_Ex2: "g ∈ Ex2"
  shows "f ∘ g ∈ Ex2"
proof -
  from f_Ex2 obtain f_1
    where f_1_Ex1: "f_1 ∈ Ex1" and "f = CONJ f_1"
    using mem_Ex2_rule by auto;
  moreover
  from g_Ex2 obtain g_1
    where g_1_Ex1: "g_1 ∈ Ex1" and "g = CONJ g_1"
    using mem_Ex2_rule by auto;
  ultimately
  have "f ∘ g = (CONJ f_1) ∘ (CONJ g_1)" by auto;
  hence "f ∘ g = CONJ (f_1 ∘ g_1)" using comp_CONJ by auto;
  moreover
  have "f_1 ∘ g_1 ∈ Ex1" using closed_comp and f_1_Ex1 and g_1_Ex1 by auto;
  ultimately
  show "f ∘ g ∈ Ex2" using mem_Ex2_rule by auto;
qed;

```

```

interpretation CofinitaryGroup Ex2;
proof;
  show "Ex2  $\subseteq$  S_inf"
  proof;
    fix f
    assume "f  $\in$  Ex2"
    with mem_Ex2_rule obtain g where "g  $\in$  Ex1" and "f = CONJ g" by auto;
    with type_CONJ show "f  $\in$  S_inf" by auto;
  qed;
next
  from id_Ex2 show "id  $\in$  Ex2" .;
next
  fix f g
  assume "f  $\in$  Ex2  $\wedge$  g  $\in$  Ex2"
  with comp_Ex2 show "f  $\circ$  g  $\in$  Ex2"; by auto;
next
  fix f
  assume "f  $\in$  Ex2"
  with inv_Ex2 show "inv f  $\in$  Ex2" by auto;
next;
  fix f
  assume "f  $\in$  Ex2  $\wedge$  f  $\neq$  id"
  with Ex2_cofinitary have "Fix f = {}" by auto;
  thus "finite (Fix f)" using finite_def by auto;
qed;

```

9 The Conclusion

With all that we have shown we have already clearly shown $Ex2$ to be a cofinitary group. The formalization also shows this, we just have to refer to the correct theorems proved above.

interpretation *CofinitaryGroup Ex2*

proof

show $Ex2 \subseteq S\text{-inf}$

proof

fix f

assume $f \in Ex2$

with *mem-Ex2-rule* obtain g where $g \in Ex1$ and $f = CONJ\ g$ by *auto*

with *type-CONJ* show $f \in S\text{-inf}$ by *auto*

qed

next

from *id-Ex2* show $id \in Ex2$.

next

fix $f\ g$

assume $f \in Ex2 \wedge g \in Ex2$

with *comp-Ex2* show $f \circ g \in Ex2$ by *auto*

next

fix f

assume $f \in Ex2$

with *inv-Ex2* show $inv\ f \in Ex2$ by *auto*

next

fix f

assume $f \in Ex2 \wedge f \neq id$

with *Ex2-cofinitary* have $Fix\ f = \{\}$ by *auto*

thus *finite* ($Fix\ f$) using *finite-def* by *auto*

qed

end

References

Isabelle: <http://isabelle.in.tum.de>

Has the software for download, installation instructions, documentation, links to more information.

Proof General: <http://proofgeneral.inf.ed.ac.uk/>

Bart Kastermans, An Example of a Cofinitary Group in Isabelle/HOL, www.bartk.nl/files.php