An Example of a Cofinitary Group in Isabelle/HOL

Bart Kastermans

June 11, 2019

Abstract

We formalize the usual proof that the group generated by the function \( k \mapsto k + 1 \) on the integers gives rise to a cofinitary group.

Contents

1 Introduction 1
2 The Main Notions 3
3 The Function \( upOne \) 4
4 The Set of Functions and Normal Forms 5
5 All Elements Cofinitary Bijections. 7
6 Closed under Composition and Inverse 8
7 Conjugation with a Bijection 11
8 Bijections on \( \mathbb{N} \) 12
9 The Conclusion 15

theory CofGroups
imports Main HOL~Library.Nat-Bijection
begin

1 Introduction

Cofinitary groups have received a lot of attention in Set Theory. I will start by giving some references, that together give a nice view of the area. See also Kastermans [7] for my view of where the study of these groups (other
than formalization) is headed. Starting work was done by Adeleke [1], Truss [12] and [13], and Koppelberg [10]. Cameron [3] is a very nice survey. There is also work on cardinal invariants related to these groups and other almost disjoint families, see e.g. Brendle, Spinas, and Zhang [2], Hrušák, Steprans, and Zhang [5], and Kastermans and Zhang [9]. Then there is also work on constructions and descriptive complexity of these groups, see e.g. Zhang [14], Gao and Zhang [4], and Kastermans [6] and [8].

In this note we work through formalizing a basic example of a cofinitary group. We want to achieve two things by working through this example. First how to formalize some proofs from basic set-theoretic algebra, and secondly, to do some first steps in the study of formalization of this area of set theory. This is related to the work of Paulson and Grabczewski [11] on formalizing set theory, our preference however is towards using Isar resulting in a development more readable for “normal” mathematicians.

A cofinitary group is a subgroup $G$ of the symmetric group on $\mathbb{N}$ (in Isabelle $\text{nat}$) such that all non-identity elements $g \in G$ have finitely many fixed points. A simple example of a cofinitary group is obtained by considering the group $G'$ a subgroup of the symmetric group on $\mathbb{Z}$ (in Isabelle $\text{int}$ generated by the function $\text{upOne} : \mathbb{Z} \to \mathbb{Z}$ defined by $k \mapsto k + 1$. No element in this group other than the identity has a fixed point. Conjugating this group by any bijection $\mathbb{Z} \to \mathbb{N}$ gives a cofinitary group.

We will develop a workable definition of a cofinitary group (Section 2) and show that the group as described in the previous paragraph is indeed cofinitary (this takes the whole paper, but is all pulled together in Section 9). Note: formalizing the previous paragraph is all that is completed in this note.

Since this note is also written to be read by the proverbial “normal” mathematician we will sometimes remark on notations as used in Isabelle as they related to common notation. We do expect this proverbial mathematician to be somewhat flexible though. He or she will need to be flexible in reading, this is just like reading any other article; part of reading is reconstructing.

We end this introduction with a quick overview of the paper. In Section 2 we define the notion of cofinitary group. In Section 3 we define the function $\text{upOne}$ and give some of its basic properties. In Section 4 we define the set $\text{Ex1}$ which is the underlying set of the group generated by $\text{upOne}$, there we also derive a normal form theorem for the elements of this set. In Section 5 we show all elements in $\text{Ex1}$ are cofinitary bijections (cofinitary here is used in the general meaning of having finitely many fixed points). In Section 6 we show this set is closed under composition and inverse, in effect showing that it is a “cofinitary group” (cofinitary group here is in quotes, since we only define it for sets of permutations on the natural numbers). In Section 7 we show the general theorem that conjugating a permutation by a bijection
does the expected thing to the set of fixed points. In Section 8 we define the function \textsc{Conj} that is conjugation by \textit{ni-bij} (a bijection from \textit{nat} to \textit{int}), show that it acts well with respect to the group operations, use it to define \textit{Ex2} which is the underlying set of the cofinitary group we are construction, and show the basic properties of \textit{Ex2}. Finally in Section 9 we quickly show that all the work in the section before it combines to show that \textit{Ex2} is a cofinitary group.

2 The Main Notions

First we define the two main notions.

We write \textit{S-inf} for the symmetric group on the natural numbers (we do not define this as a group, only as the set of bijections).

\textbf{definition} \textit{S-inf} :: (\textit{nat} \Rightarrow \textit{nat}) set where \textit{S-inf} = \{f::(\textit{nat} \Rightarrow \textit{nat}). \textit{bij} f\}

Note here that \textit{bij} \textit{f} is the predicate that \textit{f} is a bijection. This is common notation in Isabelle, a predicate applied to an object. Related to this \textit{inj} \textit{f} means \textit{f} is injective, and \textit{surj} \textit{f} means \textit{f} is surjective.

The same notation is used for function application. Next we define a function \textit{Fix}, applying it to an object is also written by juxtaposition.

Given any function \textit{f} we define \textit{Fix f} to be the set of fixed points for this function.

\textbf{definition} \textit{Fix} :: (\textit{'a} \Rightarrow \textit{'a}) \Rightarrow (\textit{'a} set) where \textit{Fix f} = \{ n . f(n) = n \}

We then define a locale \textit{CofinitaryGroup} that represents the notion of a cofinitary group. An interpretation is given by giving a set of functions \textit{nat} \rightarrow \textit{nat} and showing that it satisfies the identities the locale assumes. A locale is a way to collect together some information that can then later be used in a flexible way (we will not make a lot of use of that here).

\textbf{locale} \textit{CofinitaryGroup} =

\textbf{fixes} \quad \textit{dom} :: (\textit{nat} \Rightarrow \textit{nat}) set

\textbf{assumes}

\textit{type-dom} : \textit{dom} \subseteq \textit{S-inf} and \textit{id-com} : \textit{id} \in \textit{dom} and \textit{mult-closed} : \textit{f} \in \textit{dom} \land \textit{g} \in \textit{dom} \Longrightarrow \textit{f} \circ \textit{g} \in \textit{dom} and \textit{inv-closed} : \textit{f} \in \textit{dom} \Longrightarrow \textit{inv} \textit{f} \in \textit{dom} and \textit{cofinitary} : \textit{f} \in \textit{dom} \land \textit{f} \neq \textit{id} \Longrightarrow \textit{finite} (\textit{Fix f})
3 The Function \textit{upOne}

Here we define the function, \textit{upOne}, translation up by 1 and proof some of its basic properties.

\textbf{definition} \textit{upOne} :: \texttt{int} \Rightarrow \texttt{int}
\textbf{where}
\textit{upOne} \texttt{n} = \texttt{n} + 1

\textbf{declare} \textit{upOne-def} \[ \texttt{simp} \] — automated tools can use the definition

First we show that this function is a bijection. This is done in the usual two parts; we show it is injective by showing from the assumption that outputs on two numbers are equal that these two numbers are equal. Then we show it is surjective by finding the number that maps to a given number.

\textbf{lemma} \textit{inj-upOne}: \textit{inj upOne}
\textbf{by} (rule \textit{Fun.injI}, \texttt{simp})

\textbf{lemma} \textit{surj-upOne}: \textit{surj upOne}
\textbf{proof} (unfold \textit{Fun.surj-def}, rule)
\textbf{fix} \texttt{k}::\texttt{int}
\textbf{show} \exists \texttt{m}. \texttt{k} = \textit{upOne} \texttt{m}
\textbf{by} (rule \textit{exI}[of \lambda l. \texttt{k} = \textit{upOne} \texttt{l} \texttt{k} − 1], \texttt{simp})
\textbf{qed}

\textbf{theorem} \textit{bij-upOne}: \textit{bij upOne}
\textbf{by} (unfold \textit{bij-def}, rule conjI \[ \textit{OF inj-upOne surj-upOne} \])

Now we show that the set of fixed points of \textit{upOne} is empty. We show this in two steps, first we show that no number is a fixed point, and then derive from this that the set of fixed points is empty.

\textbf{lemma} \textit{no-fix-upOne}: \textit{upOne} \texttt{n} \neq \texttt{n}
\textbf{proof} (rule \texttt{notI})
\textbf{assume} \textit{upOne} \texttt{n} = \texttt{n}
\textbf{with} \textit{upOne-def} \textbf{have} \texttt{n} + 1 = \texttt{n} \textbf{by simp}
\textbf{thus} \texttt{False} \textbf{by auto}
\textbf{qed}

\textbf{theorem} \textit{Fix upOne} = \{\}
\textbf{proof} –
\textbf{from} \textit{Fix-def}[of \textit{upOne}]
\textbf{have} \textit{Fix upOne} = \{\texttt{n} . \textit{upOne} \texttt{n} = \texttt{n}\} \textbf{by auto}
\textbf{with} \textit{no-fix-upOne} \textbf{have} \textit{Fix upOne} = \{\texttt{n} . \texttt{False}\} \textbf{by auto}
\textbf{with} \textit{Set.empty-def} \textbf{show} \textit{Fix upOne} = \{\} \textbf{by auto}
\textbf{qed}

Finally we derive the equation for the inverse of \textit{upOne}. The rule we use references \textit{Hilbert-Choice} since the \texttt{inv} operator, the operator that gives an inverse of a function, is defined using Hilbert’s choice operator.
lemma \( \text{inv-upOne-eq} \): \((\text{inv upOne}) (n::int) = n - 1\)

proof

\[ \text{fix } n :: \text{int} \]

have \(((\text{inv upOne}) \circ \text{upOne}) (n - 1) = (\text{inv upOne}) n\) by simp

with \text{inj-upOne} and Hilbert-Choice.inv-o-cancel

show \((\text{inv upOne}) n = n - 1\) by auto

qed

We can also show this quickly using Hilbert\_Choice.inv\_f\_eq properly instantiated:

\(\text{upOne} (n - 1) = n \Rightarrow (\text{inv upOne}) n = n - 1\).

lemma \((\text{inv upOne}) n = n - 1\)

by (rule Hilbert-Choice.inv-f-eq[of upOne n - 1 n, OF inj-upOne], simp)

\[4\] The Set of Functions and Normal Forms

We define the set \(Ex1\) of all powers of \(\text{upOne}\) and study some of its properties, note that this is the group generated by \(\text{upOne}\) (in Section 6 we prove it closed under composition and inverse). In Section 5 we show that all its elements are cofinitary and bijections (bijections with finitely many fixed points). Note that this is not a cofinitary group, since our definition requires the group to be a subset of \(S\)-inf

inductive-set \(Ex1 :: (\text{int } \Rightarrow \text{int}) \text{ set where}\)

base-func: \(\text{upOne} \in Ex1\)

comp-func: \(f \in Ex1 \Longrightarrow (\text{upOne} \circ f) \in Ex1\)

comp-inv: \(f \in Ex1 \Longrightarrow ((\text{inv upOne}) \circ f) \in Ex1\)

We start by showing a normal form for elements in this set.

lemma \(Ex1\)-Normal-form-part1: \(f \in Ex1 \Longrightarrow \exists k. \forall n. f(n) = n + k\)

proof

rule \(Ex1\).induct [of \(f\)], blast

— blast takes care of the first goal which is formal noise

assume \(f \in Ex1\)

have \(\forall n. \text{upOne} n = n + 1\) by simp

with HOL.ex1 show \(\exists k. \forall n. \text{upOne} n = n + k\) by auto

next

fix \(fa :: \text{int } \Rightarrow \text{int}\)

assume \(fa-k: \exists k. \forall n. fa n = n + k\)

thus \(\exists k. \forall n. (\text{upOne} \circ fa) n = n + k\) by auto

next

fix \(fa :: \text{int } \Rightarrow \text{int}\)

assume \(fa-k: \exists k. \forall n. fa n = n + k\)

from \(\text{inv-upOne-eq}\) have \(\forall n. (\text{inv upOne}) n = n - 1\) by auto

with \(fa-k\) show \(\exists k. \forall n. (\text{inv upOne} \circ fa) n = n + k\) by auto

qed

Now we’ll show the other direction. Then we apply rule \(\text{int-induct}\) which allows us to do the induction by first showing it true for \(k = 1\), then showing
that if true for $k = i$ it is also true for $k = i + 1$ and finally showing that if true for $k = i$ then it is also true for $k = i - 1$.

All proofs are fairly straightforward and use extensionality for functions. In the base case we are just dealing with $\text{upOne}$. In the other cases we define the function $\tilde{h}$ which satisfies the induction hypothesis. Then $f$ is obtained from this by adding or subtracting one pointwise.

In this proof we use some pattern matching to save on writing. In the statement of the theorem, we match the theorem against $\tilde{P} k$ thereby defining the predicate $\tilde{P}$.

lemma Ex1-Normal-form-part2:

$(\forall f. (\forall n. f n = n + k \longrightarrow f \in \text{Ex1})) \text{ is } \tilde{P} k$

proof (rule int-induct [of $\tilde{P} 1$])

show $\forall f. (\forall n. f n = n + 1) \longrightarrow f \in \text{Ex1}$

proof

fix $f :: \text{int} \Rightarrow \text{int}$

show $(\forall n. f n = n + 1) \longrightarrow f \in \text{Ex1}$

proof

assume $\forall n. f n = n + 1$

hence $\forall n. f n = \text{upOne} n$ by auto

with fun-eq-iff [of $f \text{upOne}, \text{THEN} \text{sym}$]

have $f = \text{upOne} by auto$

with Ex1.base-func show $f \in \text{Ex1}$ by auto

qed

next

fix $i :: \text{int}$

assume $1 \leq i$

assume induct-hyp: $\forall f. (\forall n. f n = n + i) \longrightarrow f \in \text{Ex1}$

show $\forall f. (\forall n. f n = n + (i + 1)) \longrightarrow f \in \text{Ex1}$

proof

fix $f :: \text{int} \Rightarrow \text{int}$

show $(\forall n. f n = n + (i + 1)) \longrightarrow f \in \text{Ex1}$

proof

assume $f$-eq: $\forall n. f n = n + (i + 1)$

let $\tilde{h} = \lambda n. n + i$

from induct-hyp have $h$-$\text{Ex1}: h \in \text{Ex1}$ by auto

from $f$-eq have $\forall n. f n = \text{upOne} (\tilde{h} n)$ by (unfold $\text{upOne-def,auto}$)

hence $\forall n. f n = (\text{upOne} \circ \tilde{h}) n$ by auto

with fun-eq-iff [THEN $\text{sym}, \text{of} f \text{upOne} \circ \tilde{h}]$

have $f = \text{upOne} \circ \tilde{h}$ by auto

with $h$-$\text{Ex1}$ and Ex1.comp-func[of $\tilde{h}$] show $f \in \text{Ex1}$ by auto

 qed

next

fix $i :: \text{int}$

assume $i \leq 1$

assume induct-hyp: $\forall f. (\forall n. f n = n + i) \longrightarrow f \in \text{Ex1}$


show $\forall f. (\forall n. f n = n + (i - 1)) \rightarrow f \in Ex1$

proof
  fix $f :: \text{int} \Rightarrow \text{int}$
  show $(\forall n. f n = n + (i - 1)) \rightarrow f \in Ex1$
  proof
    assume $f$-eq: $\forall n. f n = n + (i - 1)$
    let $\?h = \lambda n. n + i$
    from induct-hyp have $h$-Ex1: $\?h \in Ex1$ by auto
    from inv-upOne-eq and $f$-eq
    have $\forall n. f n = (\text{inv upOne} \circ ?h) n$ by auto
    hence $\forall n. f n = (\text{inv upOne} \circ ?h) n$ by auto
    with fun-eq-iff THEN $\text{sgm, of f}$ $\text{inv upOne} \circ ?h]$
    have $f = \text{inv upOne} \circ ?h$ by auto
    with $h$-Ex1 and $Ex1$-comp-inv[of $\?h$]
    show $f \in Ex1$ by auto
  qed
  qed
  qed

Combining the two directions we get the normal form theorem.

theorem $Ex1$-Normal-form: $(f \in Ex1) = (\exists k. \forall n. f(n) = n + k)$
proof
  assume $f \in Ex1$
  with $Ex1$-Normal-form-part1 [of $f$]
  show $(\exists k. \forall n. f(n) = n + k)$ by auto
next
  assume $\exists k. \forall n. f(n) = n + k$
  with $Ex1$-Normal-form-part2
  show $f \in Ex1$ by auto
  qed

5 All Elements Cofinitary Bijections.

We now show all elements in $\text{CofGroups.Ex1}$ are bijections, Theorem $all$-bij, and have no fixed points, Theorem no-fixed-pt.

theorem $all$-bij: $f \in Ex1 \implies \text{bij } f$
proof (unfold bij-def)
  assume $f \in Ex1$
  with $Ex1$-Normal-form
  obtain $k$ where $f$-eq: $\forall n. f n = n + k$ by auto

  show $\text{inj } f \land \text{surj } f$
  proof (rule conjI)
    show INJ: inj $f$
    proof (rule injI)
      fix $n m$
      assume $f n = f m$
      with $f$-eq have $n + k = m + k$ by auto
      thus $n = m$ by auto
qed

next

show \textit{SURJ}: \textit{surj} \( f \)
proof (unfold \textit{Fun}, \textit{surj-def}, rule allI)
fix \( n \)
from \( f \)-eq have \( n = f \ (n - k) \) by auto
thus \( \exists m. \ n = f \ m \) by (rule exI)
qed

theorem \textit{no-fixed-pt}:
assumes \( f\text{-Ex1} \): \( f \in \textit{Ex1} \)
and \( f\text{-not-id} \): \( f \neq \text{id} \)
shows \( \text{Fix} \ f = \{\} \)
proof
— we start by proving an easy general fact
have \( f\text{-eq-then-id} \): \( (\forall n. \ f(n) = n) \implies f = \text{id} \)
proof
— assume \( f\text{-prop} \): \( \forall n. \ f(n) = n \)
have \( (f x = id x) = (f x = x) \) by simp
hence \( (\forall x. \ (f x = id x)) = (\forall x. \ (f x = x)) \) by simp
with \textit{fun-eq-iff}[THEN sym, of \( f \ \text{id} \)] and \( f\text{-prop} \) show \( f = \text{id} \) by auto
qed
from \( f\text{-Ex1} \) and \( \textit{Ex1-Normal-form} \) have \( \exists k. \ (\forall n. \ f(n) = n + k) \) by auto
then obtain \( k \) where \( k\text{-prop} : \forall n. \ f(n) = n + k \) .
hence \( k = 0 \implies \forall n. \ f(n) = n \) by auto
with \( f\text{-eq-then-id} \) and \( f\text{-not-id} \) have \( k \neq 0 \) by auto
with \( k\text{-prop} \) have \( \forall n. \ f(n) \neq n \) by auto
moreover
from \( \text{Fix-def}[\text{of \( f \)}] \) have \( \text{Fix} \ f = \{n . \ f(n) = n\} \) by auto
ultimately have \( \text{Fix} \ f = \{n. \ \text{False}\} \) by auto
with Set.empty-def show \( \text{Fix} \ f = \{\} \) by auto
qed

6 Closed under Composition and Inverse

We start by showing that this set is closed under composition. These facts can later be conjugated to easily obtain the corresponding results for the group on the natural numbers.

theorem \textit{closed-comp}:
\( f \in \textit{Ex1} \land g \in \textit{Ex1} \implies f \circ g \in \textit{Ex1} \)
proof (rule \textit{Ex1.induct [of \( f \)}, blast)
assume \( f \in \textit{Ex1} \land g \in \textit{Ex1} \)
with \( \textit{Ex1-comp-func}[\text{of \( g \)}] \) show \( \text{upOne} \circ g \in \textit{Ex1} \) by auto
next
fix \( fa \)
bterm{assume}{fa \circ g \in \textit{Ex1}
with Ex1.comp-func [of fa o g]
and Fun.o-assoc [of upOne fa g]
show upOne o fa o g ∈ Ex1 by auto

next
fix fa
assume fa o g ∈ Ex1
with Ex1.comp-inv [of fa o g]
and Fun.o-assoc [of inv upOne fa g]
show (inv upOne) o fa o g ∈ Ex1 by auto
qed

now we show the set is closed under inverses. this is done by an induction
on the definition of CofGroups.Ex1 only using the normal form theorem and
rewriting of expressions.

theorem closed-inv: f ∈ Ex1 ===> inv f ∈ Ex1
proof (rule Ex1.induct [of f], blast)
assume f ∈ Ex1
show inv upOne ∈ Ex1 (is ?right ∈ Ex1)
proof –
let ?left = inv upOne o (inv upOne o upOne)
{from Ex1.comp-inv and Ex1.base-func have ?left ∈ Ex1 by auto}
moreover
{from bij-upOne and bij-is-inj have inj upOne by auto
  hence inv upOne o upOne = id by auto
  hence ?left = ?right by auto}
ultimately
show ?thesis by auto
qed

next
fix f
assume f-Ex1: f ∈ Ex1
from f-Ex1 and Ex1.Normal-form
obtain k where f-eq: ∀ n. f n = n + k by auto

show inv (upOne o f) ∈ Ex1
proof –
let ?ic = inv (upOne o f)
let ?ci = inv f o inv upOne
{— first we get an expression for inv f o inv upOne
  {from all-bij and f-Ex1 have bij f by auto
  with bij-is-inj have inj-f: inj f by auto
  have ∀ n. inv f n = n − k
  proof
fix n
from f-eq have \( f(n - k) = n \) by auto
with inv-f-eq[of f n-k n] and inj-f
show \( inv f n = n-k \) by auto
qed
with inv-upOne-eq
have \( \forall n. \?ci n = n - k - 1 \) by auto
hence \( \forall n. \?ci n = n + (-1 - k) \) by arith
}
moreover
— then we check that this implies \( inv f \circ inv upOne \) is
— a member of CofGroups.Ex1
{
from Ex1-Normal-form-part2[of -1 - k]
have \( \forall f. ((\forall n. f n = n + (-1 - k)) \rightarrow f \in Ex1) \) by auto
}
ultimately
have \( \?ci \in Ex1 \) by auto
}
moreover
{
from f-Ex1 all-bij have bij f by auto
with bij-upOne and o-inv-distrib[THEN sym]
have \( \?ci = \?ic \) by auto
}
ultimately show \( \?thesis \) by auto
qed
next
fix f
assume f-Ex1: f \in Ex1
with Ex1-Normal-form
obtain k where f-eq: \( \forall n. f n = n + k \) by auto

show \( inv (inv upOne \circ f) \in Ex1 \)
proof —
let \( \?ic = inv (inv upOne \circ f) \)
let \( \?c = inv f \circ upOne \)
{
from all-bij and f-Ex1 have bij f by auto
with bij-is-inj have inj-f: inj f by auto
have \( \forall n. inv f n = n - k \)

proof
fix n
from f-eq have \( f(n - k) = n \) by auto
with inv-f-eq[of f n-k n] and inj-f
show \( inv f n = n-k \) by auto
qed
with upOne-def
have \( \forall n. (inv f \circ upOne) n = n - k + 1 \) by auto
hence $\forall n. (\text{inv } f \circ \text{upOne}) n = n + (1 - k)$ by arith
moreover
from \textit{Ex1-Normal-form-part2}[of 1 - k]
have $(\forall f. (\forall n. f \circ n = n + (1 - k)) \implies f \in \text{Ex1})$ by auto
ultimately
have $?c \in \text{Ex1}$ by auto
}
moreover
{
from f-Ex1 all-bij and bij-is-inj have bij f by auto
moreover
from bij-upOne and bij-imp-bij-inv have bij (inv upOne) by auto
moreover
note o-inv-distrib[THEN sym]
ultimately
have inv f $\circ$ inv (inv upOne) = inv (inv upOne $\circ$ f) by auto
moreover
from bij-upOne and inv-inv-eq
have inv (inv upOne) = upOne by auto
ultimately
have $?c = ?ic$ by auto
}
ultimately
show $\text{thesis}$ by auto
qed
qed

7 Conjugation with a Bijection

An abbreviation of the bijection from the natural numbers to the integers defined in the library. This will be used to coerce the functions above to be on the natural numbers.

abbreviation ni-bij $==$ int-decode

lemma bij-f-o-inf-f: bij f $\implies$ f $\circ$ inv f = id
unfolding bij-def surj-iff by simp

The following theorem is a key theorem in showing that the group we are interested in is cofinitary. It states that when you conjugate a function with a bijection the fixed points get mapped over.

theorem conj-fix-pt: $\forall f::('a => 'b), \forall g::('b => 'b), (bij f)$
$\implies$ \((\text{inv } f) \cdot (\text{Fix } g)) = \text{Fix } (\text{inv } f \circ g \circ f)$
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 \((\text{inv } f)'(\text{Fix } g)\) = \text{Fix} \((\text{inv } f) \circ g \circ f\)

thm \text{set-eq-subset[of} \((\text{inv } f)'(\text{Fix } g)\) \text{Fix}((\text{inv } f) \circ g \circ f)]\)

proof
show \((\text{inv } f)'(\text{Fix } g)\) \subseteq \text{Fix} \((\text{inv } f) \circ g \circ f\)
proof
fix \(x\)
assume \(x \in (\text{inv } f)'(\text{Fix } g)\)
with \text{image-def} have \(\exists y \in \text{Fix } g. \ x = (\text{inv } f) \ y\) by auto
from \text{this} obtain \(y\) where \(y\)-prop: \(y \in \text{Fix } g \land x = (\text{inv } f) \ y\) by auto
hence \(x = (\text{inv } f) \ y\) ..
hence \(f \ x = (f \circ \text{inv } f) \ y\) by auto
with \text{bij-f} and \text{bij-f-o-inf-f[of } f\] have \(f\)-x-y: \(f \ x = y\) by auto
hence \((\text{inv } f) \ (g \ (f \ x))) = x\) by auto
with \text{inv-f-f} and \text{inj-f} have \((\text{inv } f) \ (g \ (f \ x))) = x\) by auto
hence \((\text{inv } f) \circ g \circ f) \ x = x\) by auto
with \text{Fix-def[of } f \circ g \circ f\] show \(x \in \text{Fix} \ ((\text{inv } f) \circ g \circ f)\) by auto
qed

next
show \(\text{Fix} \ (\text{inv } f \circ g \circ f) \subseteq (\text{inv } f)'(\text{Fix } g)\)
proof
fix \(x\)
assume \(x \in \text{Fix} \ (\text{inv } f \circ g \circ f)\)
with \text{Fix-def[of } \text{inv } f \circ g \circ f\] have \(x\)-fix: \((\text{inv } f \circ g \circ f) \ x = x\) by auto
hence \((\text{inv } f) \ (g (f (x)))) = x\) by auto
hence \(\exists y. \ (\text{inv } f) \ y = x\) by auto
from \text{this} obtain \(y\) where \(x\)-inf-f-y: \(x = (\text{inv } f) \ y\) by auto
with \(x\)-fix have \((\text{inv } f \circ g \circ f)(\text{inv } f) \ y) = (\text{inv } f) \ y\) by auto
hence \((f \circ \text{inv } f \circ g \circ f \circ \text{inv } f) \ y) = (f \circ \text{inv } f)(y)\) by auto
with \text{o-assoc}

have \((f \circ \text{inv } f) \circ g \circ (f \circ \text{inv } f)) \ y = (f \circ \text{inv } f) \ y\) by auto
with \text{bij-f} and \text{bij-f-o-inf-f[of } f\] have \(g \ y = y\) by auto
with \text{Fix-def[of } \text{of } f\] have \(y \in \text{Fix } g\) by auto
with \(x\)-inf-f-y show \(x \in (\text{inv } f)'(\text{Fix } g)\) by auto
qed

qed

8 Bijective on \(\mathbb{N}\)

In this section we define the subset \(\text{Ex2}\) of \(\text{S-inf}\) that is the conjugate of \(\text{CofGroups.Ex1 \ bij \ ni-bij}\), and show its basic properties.

\(\text{CONJ}\) is the function that will conjugate \(\text{CofGroups.Ex1}\) to \(\text{Ex2}\).
definition \textit{CONJ} :: (int \Rightarrow int) \Rightarrow (nat \Rightarrow nat)

where
\textit{CONJ} f = (\textit{inv ni-bij} ) \circ f \circ \textit{ni-bij}

declare \textit{CONJ-def} \ [simp] — automated tools can use the definition

We quickly check that this function is of the right type, and then show three of its properties that are very useful in showing \textit{Ex2} is a group.

\textbf{lemma} \textit{type-CONJ}: \( f \in \textit{Ex1} \implies (\textit{inv ni-bij}) \circ f \circ \textit{ni-bij} \in S\text{-inf} \)
\textbf{proof} —
\begin{itemize}
\item assume \( f\text{-Ex1}: f \in \textit{Ex1} \)
\item with all-bij have \( \textit{bij f} \) by auto
\item with bij-int-decode and bij-comp
\item have \( \textit{bij-f-nibij}: \textit{bij} (f \circ \textit{ni-bij}) \) by auto
\item with bij-int-decode and bij-imp-bij-inv have \( \textit{bij} (\textit{inv ni-bij}) \) by auto
\item with bij-f-nibij and bij-comp[af \( f \circ \textit{ni-bij} \textit{inv ni-bij} \)]
\item and o-assoc[af \( \textit{inv ni-bij} f \textit{ni-bij} \)]
\item have \( \textit{bij} ((\textit{inv ni-bij}) \circ f \circ \textit{ni-bij}) \) by auto
\item with S-inf-def show \( ((\textit{inv ni-bij}) \circ f \circ \textit{ni-bij}) \in S\text{-inf} \) by auto
\end{itemize}
qed

\textbf{lemma} \textit{inv-CONJ}:
\begin{itemize}
\item assumes \( \textit{bij-f} \)
\item shows \( \textit{inv} (\textit{CONJ f}) = \textit{CONJ} (\textit{inv f}) \) (is ?left = ?right)
\end{itemize}
\textbf{proof} —
\begin{itemize}
\item have \( \textit{st1}: \textit{?left} = \textit{inv} ((\textit{inv ni-bij}) \circ f \circ \textit{ni-bij}) \)
\item using \textit{CONJ-def} by auto
\item from bij-int-decode and bij-imp-bij-inv
\item have \( \textit{inv-ni-bij-bij}: \textit{bij} (\textit{inv ni-bij}) \) by auto
\item with bij-f and bij-comp have \( \textit{bij} (\textit{inv ni-bij} \circ f) \) by auto
\item with o-inv-distrib[af \( \textit{inv ni-bij} \circ f \textit{ni-bij} \)] and bij-int-decode
\item have \( \textit{inv} ((\textit{inv ni-bij}) \circ f \circ \textit{ni-bij}) =
\item (\textit{inv ni-bij}) \circ (\textit{inv} ((\textit{inv ni-bij}) \circ f)) \) by auto
\item with \( \textit{st1} \) have \( \textit{st2}: \textit{?left} =
\item ((\textit{inv ni-bij}) \circ (\textit{inv} ((\textit{inv ni-bij}) \circ f))) \) by auto
\item from \( \textit{inv-ni-bij-bij} \) and \( \textit{bij f} \) and o-inv-distrib
\item have \( \textit{h1}: \textit{inv} (\textit{inv ni-bij} \circ f) = \textit{inv} f \circ \textit{inv} (\textit{inv (ni-bij)}) \) by auto
\item from bij-int-decode and inv-inv-eq[af \( \textit{ni-bij} \)]
\item have \( \textit{inv} (\textit{inv ni-bij}) = \textit{ni-bij} \) by auto
\item with \( \textit{st2} \) and \( \textit{h1} \) have \( \textit{?left} = (\textit{inv ni-bij} \circ (\textit{inv f} \circ (\textit{ni-bij}))) \) by auto
\item with o-assoc have \( \textit{?left} = \textit{inv ni-bij} \circ \textit{inv f} \circ \textit{ni-bij} \) by auto
\item with \textit{CONJ-def}[af \( \textit{inv f} \)] show \( \textit{?thesis} \) by auto
\end{itemize}
qed

\textbf{lemma} \textit{comp-CONJ}:
\begin{itemize}
\item \( \textit{CONJ} (\textit{f} \circ \textit{g}) = (\textit{CONJ} \textit{f}) \circ (\textit{CONJ} \textit{g}) \) (is ?left = ?right)
\end{itemize}
\textbf{proof} —
\begin{itemize}
\item from bij-int-decode have surj ni-bij unfolding bij-def by auto
\end{itemize}
then have \( \text{ni-bij} \circ \text{inv ni-bij} = \text{id} \) unfolding surj-iff by auto
moreover have \(?\text{left} = (\text{inv ni-bij} \circ (f \circ g) \circ \text{ni-bij} \) by simp
hence \(?\text{left} = (\text{inv ni-bij} \circ ((f \circ \text{id}) \circ g) \circ \text{ni-bij} \) by simp
ultimately have \(?\text{left} = (\text{inv ni-bij} \circ ((f \circ (\text{ni-bij} \circ \text{inv ni-bij})) \circ g) \circ \text{ni-bij} \) by auto
— a simple computation using only associativity
— completes the proof
thus \(?\text{left} = \text{?right} \) by (auto simp add: o-assoc)
qed

lemma id-CONJ: \( \text{CONJ id} = \text{id} \)
proof (unfold \text{CONJ-def})
  from \text{bij-int-decode} have inj \text{ni-bij} using \text{bij-def} by auto
  hence inv \text{ni-bij} \circ \text{ni-bij} = \text{id} by auto
  thus \((\text{inv ni-bij} \circ \text{id}) \circ \text{ni-bij} = \text{id} \) by auto
qed

We now define the group we are interested in, and show the basic facts that
together will show this is a cofinitary group.

definition \( \text{Ex2} :: (\text{nat} \Rightarrow \text{nat}) \text{ set} \)
where
\( \text{Ex2} = \text{CONJ'}\text{Ex1} \)

theorem mem-Ex2-rule: \( f \in \text{Ex2} = (\exists g. (g \in \text{Ex1} \land f = \text{CONJ g}) \)
proof
  assume \( f \in \text{Ex2} \)
  hence \( f \in \text{CONJ'}\text{Ex1} \) using \text{Ex2-def} by auto
  from this obtain \( g \) where \( g \in \text{Ex1} \land f = \text{CONJ g} \) by blast
  thus \( \exists g. (g \in \text{Ex1} \land f = \text{CONJ g}) \) by auto
next
  assume \( \exists g. (g \in \text{Ex1} \land f = \text{CONJ g}) \)
  with \text{Ex2-def} show \( f \in \text{Ex2} \) by auto
qed

theorem Ex2-cofinitary:
  assumes \( f-\text{Ex2}: f \in \text{Ex2} \)
  and \( f-\text{nid}: f \neq \text{id} \)
  shows \( \text{Fix } f = \{ \} \)
proof –
  from \( f-\text{Ex2} \) and \( \text{mem-Ex2-rule} \)
  obtain \( g \) where \( g-\text{Ex1}: g \in \text{Ex1} \) and \( f-cg: f = \text{CONJ g} \) by auto
  with \text{id-CONJ} and \( f-\text{nid} \) have \( g \neq \text{id} \) by auto
  with \( g-\text{Ex1} \) and \( \text{no-fixed-pt[of } g \text{]} \) have \( fg-\text{empty}: \text{Fix } g = \{ \} \) by auto
  from \( \text{conj-fix-pt[of ni-bij g]} \) and \text{bij-int-decode}
  have \((\text{inv ni-bij})' (\text{Fix } g) = \text{Fix(\text{CONJ g})} \) by auto
  with \( fg-\text{empty} \) have \( \{ \} = \text{Fix (CONJ g)} \) by auto

14
with f-cg show Fix f = {} by auto
qed

lemma id-Ex2 : id ∈ Ex2
proof –
  from Ex1-Normal-form-part2[of 0] have id ∈ Ex1 by auto
  with id-CONJ and Ex2-def and mem-Ex2-rule show ?thesis by auto
qed

lemma inv-Ex2 : f ∈ Ex2 ⇒ (inv f) ∈ Ex2
proof –
  assume f ∈ Ex2
  with mem-Ex2-rule obtain g where g ∈ Ex1 and f = CONJ g by auto
  with closed-inv have inv g ∈ Ex1 by auto
  from ⟨f = CONJ g⟩ have if-iCg : inv f = inv (CONJ g) by auto
  from all-bij and ⟨g ∈ Ex1⟩ have bij g by auto
  with if-iCg and inv-CONJ have inv f = CONJ (inv g) by auto
  from ⟨g ∈ Ex1⟩ and closed-inv have inv g ∈ Ex1 by auto
  with ⟨inv f = CONJ (inv g)⟩ and mem-Ex2-rule show inv f ∈ Ex2 by auto
qed

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

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.

**interpolation CofinitaryGroup Ex2**

**proof**

show \( \text{Ex2} \subseteq \text{S-inf} \)

**proof**

fix \( f \)

assume \( f \in \text{Ex2} \)

with \( \text{mem-Ex2-rule} \) obtain \( g \in \text{Ex1} \) where \( g = \text{CONJ} f \) by auto

with \( \text{type-CONJ} \) show \( f \in \text{S-inf} \) by auto

qed

**next**

from \( \text{id-Ex2} \) show \( \text{id} \in \text{Ex2} \).

**next**

fix \( f, g \)

assume \( f \in \text{Ex2} \land g \in \text{Ex2} \)

with \( \text{comp-Ex2} \) show \( f \circ g \in \text{Ex2} \) by auto

**next**

fix \( f \)

assume \( f \in \text{Ex2} \)

with \( \text{inv-Ex2} \) show \( \text{inv} f \in \text{Ex2} \) by auto

**next**

fix \( f \)

assume \( f \in \text{Ex2} \land f \neq \text{id} \)

with \( \text{Ex2-cofinitary} \) have \( \text{Fix} f = \{\} \) by auto

thus finite \( (\text{Fix} f) \) using \( \text{finite-def} \) by auto

qed

**end**

**References**


