Cofinality and the Delta System Lemma

Pedro Sánchez Terraf*†

March 24, 2023

Abstract

We formalize the basic results on cofinality of linearly ordered sets and ordinals and Šanin's Lemma for uncountable families of finite sets. We work in the set theory framework of Isabelle/ZF, using the Axiom of Choice as needed.

Contents

*Universidad Nacional de Córdoba. Facultad de Matemática, Astronomía, Física y Computación.

†Centro de Investigación y Estudios de Matemática (CIEM-FaMAF), Conicet. Córdoba. Argentina. Supported by Secyt-UNC project 33620180100465CB. `https://cs.famaf.unc.edu.ar/~pedro/`

1 Introduction

The session we present gathers very basic results built on the set theory formalization of Isabelle/ZF [7]. In a sense, some of the material formalized here corresponds to a natural continuation of that work. This is even clearer after perusing Section 2, where notions like cardinal exponentiation are first defined, together with various lemmas that do not depend on the Axiom of Choice (AC); the same holds for the basic theory of cofinality of ordinals, which is developed in Section 3. In Section 4, (un)countability is defined and several results proved, now using AC freely; the latter is also needed to prove König's Theorem on cofinality of cardinal exponentiation. The simplest infinitary version of the Delta System Lemma (DSL, also known as the "Sunflower Lemma") due to Šanin is proved in Section 5, and it is applied to prove that Cohen posets satisfy the *countable chain condition*.

A greater part of this development was motivated by a joint project on the formalization of the ctm approach to forcing [1] by Gunther, Pagano, Steinberg, and the author. Indeed, most of the results presented here are required for the development of forcing. As it turns out, the material as formalized presently is not imported as a whole by the forcing formalization [3, 2], since the latter requires relativized versions of both the concepts and the proofs.

2 Library of basic *ZF* results

theory *ZF_Library*
  imports
    ZF-Constructible.Normal

begin

This theory gathers basic "combinatorial" results that can be proved in *ZF* (that is, without using the Axiom of Choice AC).

We begin by setting up math-friendly notation.

no_notation *oadd* (**infixl** ‹++› *65*)
no_notation *sum* (**infixr** ‹+› *65*)
notation *oadd* (**infixl** ‹+› *65*)

notation *nat* (‹ω›)
notation *csucc* (‹_$^+$› [*90*])
no_notation *Aleph* (‹$\aleph$_› [*90*] *90*)
notation *Aleph* (‹$\aleph$_›)
syntax *_ge* :: [*i*,*i*] ⇒ *o* (**infixl** ‹≥› *50*)
translations $x \geq y \rightharpoonup y \leq x$

2.1 Some minimal arithmetic/ordinal stuff

lemma *Un_leD1* : $i \cup j \leq k \Longrightarrow Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow i \leq k$
⟨*proof*⟩

lemma *Un_leD2* : $i \cup j \leq k \Longrightarrow Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow j \leq k$
⟨*proof*⟩

lemma *Un_memD1*: $i \cup j \in k \Longrightarrow Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow i \leq k$
⟨*proof*⟩

lemma *Un_memD2* : $i \cup j \in k \Longrightarrow Ord(i) \Longrightarrow Ord(j) \Longrightarrow Ord(k) \Longrightarrow j \leq k$
⟨*proof*⟩

This lemma allows to apply arithmetic simprocs to ordinal addition

lemma *nat_oadd_add*[*simp*]:
assumes $m \in \omega$ $n \in \omega$ **shows** $n + m = n$ #+ m
⟨*proof*⟩

lemma *Ord_has_max_imp_succ*:
assumes $Ord(\gamma)$ $\beta \in \gamma$ $\forall \alpha{\in}\gamma.\ \alpha \leq \beta$
shows $\gamma = succ(\beta)$
⟨*proof*⟩

lemma *Least_antitone*:
assumes
$Ord(j)$ $P(j)$ $\bigwedge i.\ P(i) \Longrightarrow Q(i)$
shows
$(\mu\ i.\ Q(i)) \leq (\mu\ i.\ P(i))$
⟨*proof*⟩

lemma *Least_set_antitone*:
$Ord(j) \Longrightarrow j{\in}A \Longrightarrow A \subseteq B \Longrightarrow (\mu\ i.\ i{\in}B) \leq (\mu\ i.\ i{\in}A)$
⟨*proof*⟩

lemma *le_neq_imp_lt*:
$x{\leq}y \Longrightarrow x{\neq}y \Longrightarrow x{<}y$
⟨*proof*⟩

Strict upper bound of a set of ordinals.

definition
str_bound :: *i*⇒*i* **where**

$str\_bound(A) \equiv \bigcup a{\in}A.\ succ(a)$

lemma *str_bound_type* [*TC*]: $\forall a{\in}A.\ Ord(a) \Longrightarrow Ord(str\_bound(A))$
⟨*proof*⟩

lemma *str_bound_lt*: $\forall a{\in}A.\ Ord(a) \Longrightarrow \forall a{\in}A.\ a < str\_bound(A)$
⟨*proof*⟩

lemma *naturals_lt_nat*[*intro*]: $n \in \omega \Longrightarrow n < \omega$
⟨*proof*⟩

The next two lemmas are handy when one is constructing some object recursively. The first handles injectivity (of recursively constructed sequences of sets), while the second is helpful for establishing a symmetry argument.

lemma *Int_eq_zero_imp_not_eq*:
assumes
$\bigwedge x\ y.\ x{\in}D \Longrightarrow y \in D \Longrightarrow x \neq y \Longrightarrow A(x) \cap A(y) = 0$
$\bigwedge x.\ x{\in}D \Longrightarrow A(x) \neq 0\ a{\in}D\ b{\in}D\ a{\neq}b$
shows
$A(a) \neq A(b)$
⟨*proof*⟩

lemma *lt_neq_symmetry*:
assumes
$\bigwedge\alpha\ \beta.\ \alpha \in \gamma \Longrightarrow \beta \in \gamma \Longrightarrow \alpha < \beta \Longrightarrow Q(\alpha,\beta)$
$\bigwedge\alpha\ \beta.\ Q(\alpha,\beta) \Longrightarrow Q(\beta,\alpha)$
$\alpha \in \gamma\ \beta \in \gamma\ \alpha \neq \beta$
$Ord(\gamma)$
shows
$Q(\alpha,\beta)$
⟨*proof*⟩

lemma *cardinal_succ_not_0*: $|A| = succ(n) \Longrightarrow A \neq 0$
⟨*proof*⟩

lemma *Ord_eq_Collect_lt*: $i{<}\alpha \Longrightarrow \{j{\in}\alpha.\ j{<}i\} = i$
— almost the same proof as *nat_eq_Collect_lt*
⟨*proof*⟩

2.2 Manipulation of function spaces

definition
Finite_to_one :: [*i*,*i*] ⇒ *i* **where**
$Finite\_to\_one(X,Y) \equiv \{f{:}X{\to}Y.\ \forall y{\in}Y.\ Finite(\{x{\in}X\ .\ f`x = y\})\}$

lemma *Finite_to_oneI*[*intro*]:
assumes $f{:}X{\to}Y\ \bigwedge y.\ y{\in}Y \Longrightarrow Finite(\{x{\in}X\ .\ f`x = y\})$
shows $f \in Finite\_to\_one(X,Y)$
⟨*proof*⟩

lemma *Finite_to_oneD*[*dest*]:
$f \in$ *Finite_to_one*$(X,Y) \Longrightarrow f{:}X \to Y$
$f \in$ *Finite_to_one*$(X,Y) \Longrightarrow y \in Y \Longrightarrow$ *Finite*$(\{x \in X\ .\ f`x = y\})$
⟨*proof*⟩

lemma *subset_Diff_Un*: $X \subseteq A \Longrightarrow A = (A - X) \cup X$ ⟨*proof*⟩

lemma *Diff_bij*:
assumes $\forall A \in F.\ X \subseteq A$ **shows** $(\lambda A \in F.\ A\text{-}X) \in bij(F, \{A\text{-}X.\ A \in F\})$
⟨*proof*⟩

lemma *function_space_nonempty*:
assumes $b \in B$
shows $(\lambda x \in A.\ b) : A \to B$
⟨*proof*⟩

lemma *vimage_lam*: $(\lambda x \in A.\ f(x))$ -`` $B = \{\ x \in A\ .\ f(x) \in B\ \}$
⟨*proof*⟩

lemma *range_fun_subset_codomain*:
assumes $h{:}B \to C$
shows *range*$(h) \subseteq C$
⟨*proof*⟩

lemma *Pi_rangeD*:
assumes $f \in Pi(A,B)$ $b \in$ *range*(f)
shows $\exists a \in A.\ f`a = b$
⟨*proof*⟩

lemma *Pi_range_eq*: $f \in Pi(A,B) \Longrightarrow$ *range*$(f) = \{f\ `\ x\ .\ x \in A\}$
⟨*proof*⟩

lemma *Pi_vimage_subset* : $f \in Pi(A,B) \Longrightarrow f$-``$C \subseteq A$
⟨*proof*⟩

lemma *apply_in_codomain_Ord*:
assumes
Ord(γ) $\gamma \neq 0$ $f{:}\ A \to \gamma$
shows
$f`x \in \gamma$
⟨*proof*⟩

lemma *range_eq_image*:
assumes $f{:}A \to B$
shows *range*$(f) = f``A$
⟨*proof*⟩

lemma *Image_sub_codomain*: $f{:}A \to B \Longrightarrow f``C \subseteq B$

⟨*proof*⟩

lemma *inj_to_Image*:
assumes
$f{:}A{\rightarrow}B$ $f \in inj(A,B)$
shows
$f \in inj(A,f``A)$
⟨*proof*⟩

lemma *inj_imp_surj*:
fixes *f b*
notes *inj_is_fun*[*dest*]
defines [*simp*]: $ifx(x) \equiv if\ x{\in}range(f)\ then\ converse(f)`x\ else\ b$
assumes $f \in inj(B,A)$ $b{\in}B$
shows $(\lambda x{\in}A.\ ifx(x)) \in surj(A,B)$
⟨*proof*⟩

lemma *fun_Pi_disjoint_Un*:
assumes $f \in Pi(A,B)$ $g \in Pi(C,D)$ $A \cap C = 0$
shows $f \cup g \in Pi(A \cup C,\ \lambda x.\ B(x) \cup D(x))$
⟨*proof*⟩

lemma *Un_restrict_decomposition*:
assumes $f \in Pi(A,B)$
shows $f = restrict(f,\ A \cap C) \cup restrict(f,\ A - C)$
⟨*proof*⟩

lemma *restrict_eq_imp_Un_into_Pi*:
assumes $f \in Pi(A,B)$ $g \in Pi(C,D)$ $restrict(f,\ A \cap C) = restrict(g,\ A \cap C)$
shows $f \cup g \in Pi(A \cup C,\ \lambda x.\ B(x) \cup D(x))$
⟨*proof*⟩

lemma *restrict_eq_imp_Un_into_Pi'*:
assumes $f \in Pi(A,B)$ $g \in Pi(C,D)$
$restrict(f,\ domain(f) \cap domain(g)) = restrict(g,\ domain(f) \cap domain(g))$
shows $f \cup g \in Pi(A \cup C,\ \lambda x.\ B(x) \cup D(x))$
⟨*proof*⟩

lemma *restrict_subset_Sigma*: $f \subseteq Sigma(C,B) \Longrightarrow restrict(f,A) \subseteq Sigma(A{\cap}C,\ B)$
⟨*proof*⟩

2.3 Finite sets

lemma *Replace_sing1*:
$[\![\ (\exists\, a.\ P(d,a)) \wedge (\forall\, y\ y'.\ P(d,y) \longrightarrow P(d,y') \longrightarrow y{=}y')\ ]\!] \Longrightarrow \exists\, a.\ \{y\ .\ x \in \{d\},\ P(x,y)\} = \{a\}$
⟨*proof*⟩
lemma *Replace_sing2*:

assumes $\forall\, a.\ \neg\ P(d,a)$
shows $\{y\ .\ x \in \{d\},\ P(x,y)\} = 0$
⟨*proof*⟩

lemma *Replace_sing3*:
assumes $\exists\, c\ e.\ c \neq e \wedge P(d,c) \wedge P(d,e)$
shows $\{y\ .\ x \in \{d\},\ P(x,y)\} = 0$
⟨*proof*⟩

lemma *Replace_Un*: $\{b\ .\ a \in A \cup B,\ Q(a,\ b)\} =$
$\{b\ .\ a \in A,\ Q(a,\ b)\} \cup \{b\ .\ a \in B,\ Q(a,\ b)\}$
⟨*proof*⟩

lemma *Replace_subset_sing*: $\exists\, z.\ \{y\ .\ x \in \{d\},\ P(x,y)\} \subseteq \{z\}$
⟨*proof*⟩

lemma *Finite_Replace*: $Finite(A) \Longrightarrow Finite(Replace(A,Q))$
⟨*proof*⟩

lemma *Finite_domain*: $Finite(A) \Longrightarrow Finite(domain(A))$
⟨*proof*⟩

lemma *Finite_converse*: $Finite(A) \Longrightarrow Finite(converse(A))$
⟨*proof*⟩

lemma *Finite_range*: $Finite(A) \Longrightarrow Finite(range(A))$
⟨*proof*⟩

lemma *Finite_Sigma*: $Finite(A) \Longrightarrow \forall\, x.\ Finite(B(x)) \Longrightarrow Finite(Sigma(A,B))$
⟨*proof*⟩

lemma *Finite_Pi*: $Finite(A) \Longrightarrow \forall\, x.\ Finite(B(x)) \Longrightarrow Finite(Pi(A,B))$
⟨*proof*⟩

2.4 Basic results on equipollence, cardinality and related concepts

lemma *lepollD*[*dest*]: $A \lesssim B \Longrightarrow \exists f.\ f \in inj(A,\ B)$
⟨*proof*⟩

lemma *lepollI*[*intro*]: $f \in inj(A,\ B) \Longrightarrow A \lesssim B$
⟨*proof*⟩

lemma *eqpollD*[*dest*]: $A \approx B \Longrightarrow \exists f.\ f \in bij(A,\ B)$
⟨*proof*⟩

declare *bij_imp_eqpoll*[*intro*]

lemma *range_of_subset_eqpoll*:

assumes $f \in inj(X,Y)$ $S \subseteq X$
shows $S \approx f$ `` S
⟨*proof*⟩

I thank Miguel Pagano for this proof.

lemma *function_space_eqpoll_cong*:
assumes
$A \approx A'$ $B \approx B'$
shows
$A \to B \approx A' \to B'$
⟨*proof*⟩

lemma *curry_eqpoll*:
fixes d $\nu 1$ $\nu 2$ κ
shows $\nu 1 \to \nu 2 \to \kappa \approx \nu 1 \times \nu 2 \to \kappa$
⟨*proof*⟩

lemma *Pow_eqpoll_function_space*:
fixes d X
notes *bool_of_o_def* [*simp*]
defines [*simp*]:$d(A) \equiv (\lambda x{\in}X.\ bool\_of\_o(x{\in}A))$
— the witnessing map for the thesis:
shows $Pow(X) \approx X \to 2$
⟨*proof*⟩

lemma *cantor_inj*: $f \notin inj(Pow(A),A)$
⟨*proof*⟩

definition
cexp :: $[i,i] \Rightarrow i$ (_$^{\uparrow}$_ [76,1] 75) **where**
$\kappa^{\uparrow\nu} \equiv |\nu \to \kappa|$

lemma *Card_cexp*: $Card(\kappa^{\uparrow\nu})$
⟨*proof*⟩

lemma *eq_csucc_ord*:
$Ord(i) \Longrightarrow i^{+} = |i|^{+}$
⟨*proof*⟩

I thank Miguel Pagano for this proof.

lemma *lesspoll_csucc*:
assumes $Ord(\kappa)$
shows $d \prec \kappa^{+} \longleftrightarrow d \lesssim \kappa$
⟨*proof*⟩

abbreviation
Infinite :: $i \Rightarrow o$ **where**
$Infinite(X) \equiv \neg\ Finite(X)$

lemma *Infinite_not_empty*: $Infinite(X) \Longrightarrow X \neq 0$
⟨*proof*⟩

lemma *Infinite_imp_nats_lepoll*:
assumes $Infinite(X)$ $n \in \omega$
shows $n \lesssim X$
⟨*proof*⟩

lemma *zero_lesspoll*: **assumes** $0<\kappa$ **shows** $0 \prec \kappa$
⟨*proof*⟩

lemma *lepoll_nat_imp_Infinite*: $\omega \lesssim X \Longrightarrow Infinite(X)$
⟨*proof*⟩

lemma *InfCard_imp_Infinite*: $InfCard(\kappa) \Longrightarrow Infinite(\kappa)$
⟨*proof*⟩

lemma *lt_surj_empty_imp_Card*:
assumes $Ord(\kappa)$ $\bigwedge\alpha.\ \alpha < \kappa \Longrightarrow surj(\alpha,\kappa) = 0$
shows $Card(\kappa)$
⟨*proof*⟩

2.5 Morphisms of binary relations

The main case of interest is in the case of partial orders.

lemma *mono_map_mono*:
assumes
$f \in mono\_map(A,r,B,s)$ $B \subseteq C$
shows
$f \in mono\_map(A,r,C,s)$
⟨*proof*⟩

lemma *ordertype_zero_imp_zero*: $ordertype(A,r) = 0 \Longrightarrow A = 0$
⟨*proof*⟩

lemma *mono_map_increasing*:
$j \in mono\_map(A,r,B,s) \Longrightarrow a \in A \Longrightarrow c \in A \Longrightarrow \langle a,c\rangle \in r \Longrightarrow \langle j`a,j`c\rangle \in s$
⟨*proof*⟩

lemma *linear_mono_map_reflects*:
assumes
$linear(\alpha,r)$ $trans[\beta](s)$ $irrefl(\beta,s)$ $f \in mono\_map(\alpha,r,\beta,s)$
$x \in \alpha$ $y \in \alpha$ $\langle f`x,f`y\rangle \in s$
shows
$\langle x,y\rangle \in r$
⟨*proof*⟩

lemma *irrefl_Memrel*: $irrefl(x,\ Memrel(x))$
⟨*proof*⟩

lemmas *Memrel_mono_map_reflects* = *linear_mono_map_reflects*
[*OF well_ord_is_linear*[*OF well_ord_Memrel*] *well_ord_is_trans_on*[*OF well_ord_Memrel*]
irrefl_Memrel]

— Same proof as Paulson's *mono_map_is_inj*
lemma *mono_map_is_inj'*:
⟦ *linear*(*A*,*r*); *irrefl*(*B*,*s*); $f \in$ *mono_map*(*A*,*r*,*B*,*s*) ⟧ $\Longrightarrow f \in$ *inj*(*A*,*B*)
⟨*proof*⟩

lemma *mono_map_imp_ord_iso_image*:
assumes
linear(α,*r*) *trans*[β](*s*) *irrefl*(β,*s*) $f \in$*mono_map*(α,*r*,β,*s*)
shows
$f \in$ *ord_iso*(α,*r*,$f``\alpha$,*s*)
⟨*proof*⟩

We introduce the following notation for strictly increasing maps between ordinals.

abbreviation
mono_map_Memrel :: [*i*,*i*] ⇒ *i* (**infixr** ‹$\to_<$› *60*) **where**
$\alpha \to_< \beta \equiv$ *mono_map*(α,*Memrel*(α),β,*Memrel*(β))

lemma *mono_map_imp_ord_iso_Memrel*:
assumes
Ord(α) *Ord*(β) $f{:}\alpha \to_< \beta$
shows
$f \in$ *ord_iso*(α,*Memrel*(α),$f``\alpha$,*Memrel*(β))
⟨*proof*⟩

lemma *mono_map_ordertype_image'*:
assumes
$X \subseteq \alpha$ *Ord*(α) *Ord*(β) $f \in$ *mono_map*(*X*,*Memrel*(α),β,*Memrel*(β))
shows
ordertype($f``X$,*Memrel*(β)) = *ordertype*(*X*,*Memrel*(α))
⟨*proof*⟩

lemma *mono_map_ordertype_image*:
assumes
Ord(α) *Ord*(β) $f{:}\alpha \to_< \beta$
shows
ordertype($f``\alpha$,*Memrel*(β)) = α
⟨*proof*⟩

lemma *apply_in_image*: $f{:}A \to B \Longrightarrow a \in A \Longrightarrow f`a \in f``A$
⟨*proof*⟩

lemma *Image_subset_Ord_imp_lt*:
assumes

$Ord(\alpha)$ $h``A \subseteq \alpha$ $x \in domain(h)$ $x \in A$ $function(h)$
shows
$h`x < \alpha$
⟨*proof*⟩

lemma *ordermap_le_arg*:
assumes
$X \subseteq \beta$ $x \in X$ $Ord(\beta)$
shows
$x \in X \Longrightarrow ordermap(X,Memrel(\beta))`x \leq x$
⟨*proof*⟩

lemma *subset_imp_ordertype_le*:
assumes
$X \subseteq \beta$ $Ord(\beta)$
shows
$ordertype(X,Memrel(\beta)) \leq \beta$
⟨*proof*⟩

lemma *mono_map_imp_le*:
assumes
$f \in mono\_map(\alpha,\ Memrel(\alpha),\beta,\ Memrel(\beta))$ $Ord(\alpha)$ $Ord(\beta)$
shows
$\alpha \leq \beta$
⟨*proof*⟩
lemmas *Memrel_mono_map_is_inj* = *mono_map_is_inj*
[*OF well_ord_is_linear*[*OF well_ord_Memrel*]
wf_imp_wf_on[*OF wf_Memrel*]]

lemma *mono_mapI*:
assumes f: $A \to B$ $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow \langle x,y \rangle \in r \Longrightarrow \langle f`x,f`y \rangle \in s$
shows $f \in mono\_map(A,r,B,s)$
⟨*proof*⟩

lemmas *mono_mapD* = *mono_map_is_fun mono_map_increasing*

bundle *mono_map_rules* = *mono_mapI*[*intro*!] *mono_map_is_fun*[*dest*] *mono_mapD*[*dest*]

lemma *nats_le_InfCard*:
assumes $n \in \omega$ $InfCard(\kappa)$
shows $n \leq \kappa$
⟨*proof*⟩

lemma *nat_into_InfCard*:
assumes $n \in \omega$ $InfCard(\kappa)$
shows $n \in \kappa$
⟨*proof*⟩

2.6 Alephs are infinite cardinals

lemma *Aleph_zero_eq_nat*: $\aleph_0 = \omega$
⟨*proof*⟩

lemma *InfCard_Aleph*:
notes *Aleph_zero_eq_nat*[*simp*]
assumes $Ord(\alpha)$
shows $InfCard(\aleph_\alpha)$
⟨*proof*⟩

Most properties of cardinals depend on AC, even for the countable. Here we just state the definition of this concept, and most proofs will appear after assuming Choice.

definition
countable :: $i \Rightarrow o$ **where**
$countable(X) \equiv X \lesssim \omega$

lemma *countableI*[*intro*]: $X \lesssim \omega \Longrightarrow countable(X)$
⟨*proof*⟩

lemma *countableD*[*dest*]: $countable(X) \Longrightarrow X \lesssim \omega$
⟨*proof*⟩

A *delta system* is family of sets with a common pairwise intersection. We will work with this notion in Section 5, but we state the definition here in order to have it available in a choiceless context.

definition
delta_system :: $i \Rightarrow o$ **where**
$delta\_system(D) \equiv \exists r.\ \forall A \in D.\ \forall B \in D.\ A \neq B \longrightarrow A \cap B = r$

lemma *delta_systemI*[*intro*]:
assumes $\forall A \in D.\ \forall B \in D.\ A \neq B \longrightarrow A \cap B = r$
shows $delta\_system(D)$
⟨*proof*⟩

lemma *delta_systemD*[*dest*]:
$delta\_system(D) \Longrightarrow \exists r.\ \forall A \in D.\ \forall B \in D.\ A \neq B \longrightarrow A \cap B = r$
⟨*proof*⟩

Hence, pairwise intersections equal the intersection of the whole family.

lemma *delta_system_root_eq_Inter*:
assumes $delta\_system(D)$
shows $\forall A \in D.\ \forall B \in D.\ A \neq B \longrightarrow A \cap B = \bigcap D$
⟨*proof*⟩

lemmas *Limit_Aleph* = *InfCard_Aleph*[*THEN InfCard_is_Limit*]

lemmas *Aleph_cont* = *Normal_imp_cont*[*OF Normal_Aleph*]

lemmas *Aleph_sup* = *Normal_Union*[*OF* _ _ *Normal_Aleph*]

bundle *Ord_dests* = *Limit_is_Ord*[*dest*] *Card_is_Ord*[*dest*]
bundle *Aleph_dests* = *Aleph_cont*[*dest*] *Aleph_sup*[*dest*]
bundle *Aleph_intros* = *Aleph_increasing*[*intro*!]
bundle *Aleph_mem_dests* = *Aleph_increasing*[*OF ltI*, *THEN ltD*, *dest*]

2.7 Transfinite recursive constructions

definition
rec_constr :: [*i*,*i*] ⇒ *i* **where**
rec_constr(*f*,α) ≡ *transrec*(α,λ*a g*. *f*‘(*g*‘‘*a*))

The function *rec_constr* allows to perform *recursive constructions*: given a choice function on the powerset of some set, a transfinite sequence is created by successively choosing some new element.

The next result explains its use.

lemma *rec_constr_unfold*: *rec_constr*(*f*,α) = *f*‘({*rec_constr*(*f*,β). β∈α})
⟨*proof*⟩

lemma *rec_constr_type*: **assumes** *f*:*Pow*(*G*)→ *G Ord*(α)
shows *rec_constr*(*f*,α) ∈ *G*
⟨*proof*⟩

end

3 Cofinality

theory *Cofinality*
imports
ZF_Library
begin

3.1 Basic results and definitions

A set X is *cofinal* in A (with respect to the relation r) if every element of A is "bounded above" by some element of X. Note that X does not need to be a subset of A.

definition
cofinal :: [*i*,*i*,*i*] ⇒ *o* **where**
cofinal(*X*,*A*,*r*) ≡ ∀ *a*∈*A*. ∃ *x*∈*X*. ⟨*a*,*x*⟩∈*r* ∨ *a* = *x*

A function is cofinal if it range is.

definition
cofinal_fun :: [*i*,*i*,*i*] ⇒ *o* **where**
cofinal_fun(*f*,*A*,*r*) ≡ ∀ *a*∈*A*. ∃ *x*∈*domain*(*f*). ⟨*a*,*f*‘*x*⟩∈*r* ∨ *a* = *f*‘*x*

lemma *cofinal_funI*:
  assumes $\bigwedge a.\ a\in A \Longrightarrow \exists x\in domain(f).\ \langle a,f`x\rangle\in r \vee a = f`x$
  shows *cofinal_fun*(f,A,r)
  ⟨*proof*⟩

lemma *cofinal_funD*:
  assumes *cofinal_fun*(f,A,r) $a\in A$
  shows $\exists x\in domain(f).\ \langle a,f`x\rangle\in r \vee a = f`x$
  ⟨*proof*⟩

lemma *cofinal_in_cofinal*:
  assumes
    trans(r) *cofinal*(Y,X,r) *cofinal*(X,A,r)
  shows
    cofinal(Y,A,r)
  ⟨*proof*⟩

lemma *codomain_is_cofinal*:
  assumes *cofinal_fun*(f,A,r) $f{:}C \to D$
  shows *cofinal*(D,A,r)
  ⟨*proof*⟩

lemma *cofinal_range_iff_cofinal_fun*:
  assumes *function*(f)
  shows *cofinal*$(range(f),A,r) \longleftrightarrow$ *cofinal_fun*(f,A,r)
  ⟨*proof*⟩

lemma *cofinal_comp*:
  assumes
    $f\in$ *mono_map*(C,s,D,r) *cofinal_fun*(f,D,r) $h{:}B \to C$ *cofinal_fun*(h,C,s)
    trans(r)
  shows *cofinal_fun*$(f\ O\ h,D,r)$
  ⟨*proof*⟩

definition
  cf_fun :: $[i,i] \Rightarrow o$ **where**
  cf_fun$(f,\alpha) \equiv$ *cofinal_fun*$(f,\alpha,Memrel(\alpha))$

lemma *cf_funI*[*intro*!]: *cofinal_fun*$(f,\alpha,Memrel(\alpha)) \Longrightarrow$ *cf_fun*(f,α)
  ⟨*proof*⟩

lemma *cf_funD*[*dest*!]: *cf_fun*$(f,\alpha) \Longrightarrow$ *cofinal_fun*$(f,\alpha,Memrel(\alpha))$
  ⟨*proof*⟩

lemma *cf_fun_comp*:
  assumes
    $Ord(\alpha)$ $f\in$ *mono_map*$(C,s,\alpha,Memrel(\alpha))$ *cf_fun*(f,α)
    $h{:}B \to C$ *cofinal_fun*(h,C,s)

shows $cf\_fun(f\ O\ h,\alpha)$
⟨*proof*⟩

definition
cf :: $i \Rightarrow i$ **where**
$cf(\gamma) \equiv \mu\ \beta.\ \exists A.\ A \subseteq \gamma \wedge cofinal(A,\gamma,Memrel(\gamma)) \wedge \beta = ordertype(A,Memrel(\gamma))$

lemma Ord_cf $[TC]$: $Ord(cf(\beta))$
⟨*proof*⟩

lemma $gamma\_cofinal\_gamma$:
assumes $Ord(\gamma)$
shows $cofinal(\gamma,\gamma,Memrel(\gamma))$
⟨*proof*⟩

lemma $cf\_is\_ordertype$:
assumes $Ord(\gamma)$
shows $\exists A.\ A \subseteq \gamma \wedge cofinal(A,\gamma,Memrel(\gamma)) \wedge cf(\gamma) = ordertype(A,Memrel(\gamma))$
(**is** $?P(cf(\gamma))$)
⟨*proof*⟩

lemma cf_fun_succ':
assumes $Ord(\beta)$ $Ord(\alpha)$ $f{:}\alpha \rightarrow succ(\beta)$
shows $(\exists x \in \alpha.\ f`x = \beta) \longleftrightarrow cf\_fun(f,succ(\beta))$
⟨*proof*⟩

lemma cf_fun_succ:
$Ord(\beta) \Longrightarrow f{:}1 \rightarrow succ(\beta) \Longrightarrow f`0 = \beta \Longrightarrow cf\_fun(f,succ(\beta))$
⟨*proof*⟩

lemma $ordertype\_0\_not\_cofinal\_succ$:
assumes $ordertype(A,Memrel(succ(i))) = 0$ $A \subseteq succ(i)$ $Ord(i)$
shows $\neg cofinal(A,succ(i),Memrel(succ(i)))$
⟨*proof*⟩

I thank Edwin Pacheco Rodríguez for the following lemma.

lemma cf_succ:
assumes $Ord(\alpha)$
shows $cf(succ(\alpha)) = 1$
⟨*proof*⟩

lemma cf_zero $[simp]$:
$cf(0) = 0$
⟨*proof*⟩

lemma $surj\_is\_cofinal$: $f \in surj(\delta,\gamma) \Longrightarrow cf\_fun(f,\gamma)$
⟨*proof*⟩

lemma cf_zero_iff: $Ord(\alpha) \Longrightarrow cf(\alpha) = 0 \longleftrightarrow \alpha = 0$

⟨*proof*⟩
lemma *cf_eq_one_iff*:
  assumes $Ord(\gamma)$
  shows $cf(\gamma) = 1 \longleftrightarrow (\exists \alpha.\ Ord(\alpha) \land \gamma = succ(\alpha))$
⟨*proof*⟩

lemma *ordertype_in_cf_imp_not_cofinal*:
  assumes
    $ordertype(A,Memrel(\gamma)) \in cf(\gamma)$
    $A \subseteq \gamma$
  shows
    $\neg\ cofinal(A,\gamma,Memrel(\gamma))$
⟨*proof*⟩

lemma *cofinal_mono_map_cf*:
  assumes $Ord(\gamma)$
  shows $\exists j \in mono\_map(cf(\gamma),\ Memrel(cf(\gamma)),\ \gamma,\ Memrel(\gamma))\ .\ cf\_fun(j,\gamma)$
⟨*proof*⟩

3.2 The factorization lemma

In this subsection we prove a factorization lemma for cofinal functions into ordinals, which shows that any cofinal function between ordinals can be "decomposed" in such a way that a commutative triangle of strictly increasing maps arises.

The factorization lemma has a kind of fundamental character, in that the rest of the basic results on cofinality (for, instance, idempotence) follow easily from it, in a more algebraic way.

This is a consequence that the proof encapsulates uses of transfinite recursion in the basic theory of cofinality; indeed, only one use is needed. In the setting of Isabelle/ZF, this is convenient since the machinery of recursion is pretty clumsy. On the downside, this way of presenting things results in a longer proof of the factorization lemma. This approach was taken by the author in the notes [8] for an introductory course in Set Theory.

To organize the use of the hypotheses of the factorization lemma, we set up a locale containing all the relevant ingredients.

locale *cofinal_factor* =
  fixes $j\ \delta\ \xi\ \gamma\ f$
  assumes *j_mono*: $j :\xi \to_< \gamma$
    and *ords*: $Ord(\delta)\ Ord(\xi)\ Limit(\gamma)$
    and *f_type*: $f\colon \delta \to \gamma$
begin

Here, f is cofinal function from δ to γ, and the ordinal ξ is meant to be the cofinality of γ. Hence, there exists an increasing map j from ξ to γ by the last lemma.

The main goal is to construct an increasing function $g \in \xi \to \delta$ such that the composition $f\ O\ g$ is still cofinal but also increasing.

definition
  factor_body :: [*i*,*i*,*i*] ⇒ *o* **where**
  $factor\_body(\beta,h,x) \equiv (x{\in}\delta \wedge j`\beta \le f`x \wedge (\forall\,\alpha{<}\beta\ .\ f`(h`\alpha) < f`x)) \vee x{=}\delta$

definition
  factor_rec :: [*i*,*i*] ⇒ *i* **where**
  $factor\_rec(\beta,h) \equiv\ \mu\ x.\ factor\_body(\beta,h,x)$

factor_rec is the inductive step for the definition by transfinite recursion of the *factor* function (called g above), which in turn is obtained by minimizing the predicate *factor_body*. Next we show that this predicate is monotonous.

lemma *factor_body_mono*:
  assumes
    $\beta{\in}\xi\ \ \alpha{<}\beta$
    $factor\_body(\beta,\lambda x{\in}\beta.\ G(x),x)$
  shows
    $factor\_body(\alpha,\lambda x{\in}\alpha.\ G(x),x)$
⟨*proof*⟩

lemma *factor_body_simp*[*simp*]: $factor\_body(\alpha,g,\delta)$
  ⟨*proof*⟩

lemma *factor_rec_mono*:
  assumes
    $\beta{\in}\xi\ \ \alpha{<}\beta$
  shows
    $factor\_rec(\alpha,\lambda x{\in}\alpha.\ G(x)) \le factor\_rec(\beta,\lambda x{\in}\beta.\ G(x))$
  ⟨*proof*⟩

We now define the factor as higher-order function. Later it will be restricted to a set to obtain a bona fide function of type *i*.

definition
  factor :: *i* ⇒ *i* **where**
  $factor(\beta) \equiv transrec(\beta,factor\_rec)$

lemma *factor_unfold*:
  $factor(\alpha) = factor\_rec(\alpha,\lambda x{\in}\alpha.\ factor(x))$
  ⟨*proof*⟩

lemma *factor_mono*:
  assumes $\beta{\in}\xi\ \ \alpha{<}\beta\ \ factor(\alpha){\neq}\delta\ \ factor(\beta){\neq}\delta$
  shows $factor(\alpha) \le factor(\beta)$
⟨*proof*⟩

The factor satisfies the predicate body of the minimization.

lemma *factor_body_factor*:

factor_body(α,$\lambda x\in\alpha$. *factor*(x),*factor*(α))
⟨*proof*⟩

lemma *factor_type* [*TC*]: *Ord*(*factor*(α))
⟨*proof*⟩

The value δ in *factor_body* (and therefore, in *factor*) is meant to be a "default value". Whenever it is not attained, the factor function behaves as expected: It is increasing and its composition with f also is.

lemma *f_factor_increasing*:
assumes $\beta\in\xi$ $\alpha<\beta$ *factor*(β)$\neq\delta$
shows *f`factor*($\alpha$) $<$ *f`factor*(β)
⟨*proof*⟩

lemma *factor_increasing*:
assumes $\beta\in\xi$ $\alpha<\beta$ *factor*(α)$\neq\delta$ *factor*(β)$\neq\delta$
shows *factor*(α)$<$*factor*(β)
⟨*proof*⟩

lemma *factor_in_delta*:
assumes *factor*(β) $\neq\delta$
shows *factor*(β) $\in\delta$
⟨*proof*⟩

Finally, we define the (set) factor function as the restriction of factor to the ordinal ξ.

definition
fun_factor :: *i* **where**
fun_factor $\equiv$ $\lambda\beta\in\xi$. *factor*(β)

lemma *fun_factor_is_mono_map*:
assumes $\bigwedge\beta$. $\beta\in\xi\Longrightarrow$ *factor*(β) $\neq\delta$
shows *fun_factor* $\in$ *mono_map*(ξ, *Memrel*(ξ), δ, *Memrel*(δ))
⟨*proof*⟩

lemma *f_fun_factor_is_mono_map*:
assumes $\bigwedge\beta$. $\beta\in\xi\Longrightarrow$ *factor*(β) $\neq\delta$
shows *f O fun_factor* $\in$ *mono_map*(ξ, *Memrel*(ξ), γ, *Memrel*(γ))
⟨*proof*⟩

end — *cofinal_factor*

We state next the factorization lemma.

lemma *cofinal_fun_factorization*:
notes *le_imp_subset* [*dest*] *lt_trans2* [*trans*]
assumes
Ord(δ) *Limit*(γ) *f*: $\delta\rightarrow\gamma$ *cf_fun*(*f*,γ)
shows

$\exists\, g \in cf(\gamma) \rightarrow_< \delta.\ \ f\ O\ g : cf(\gamma) \rightarrow_< \gamma\ \wedge$
cofinal_fun$(f\ O\ g,\gamma,$*Memrel*$(\gamma))$
⟨*proof*⟩

As a final observation in this part, we note that if the original cofinal map was increasing, then the factor function is also cofinal.

lemma *factor_is_cofinal*:
assumes
Ord(δ) *Ord*(γ)
$f : \delta \rightarrow_< \gamma$ $\ f\ O\ g \in$ *mono_map*$(\alpha,r,\gamma,$*Memrel*$(\gamma))$
cofinal_fun$(f\ O\ g,\gamma,$*Memrel*$(\gamma))$ $g\colon \alpha \rightarrow \delta$
shows
cf_fun(g,δ)
⟨*proof*⟩

3.3 Classical results on cofinalities

Now the rest of the results follow in a more algebraic way. The next proof one invokes a case analysis on whether the argument is zero, a successor ordinal or a limit one; the last case being the most relevant one and is immediate from the factorization lemma.

lemma *cf_le_domain_cofinal_fun*:
assumes
Ord(γ) *Ord*(δ) $f{:}\delta \rightarrow \gamma$ *cf_fun*(f,γ)
shows
$cf(\gamma) \leq \delta$
⟨*proof*⟩

lemma *cf_ordertype_cofinal*:
assumes
Limit(γ) $A \subseteq \gamma$ *cofinal*$(A,\gamma,$*Memrel*$(\gamma))$
shows
$cf(\gamma) = cf($*ordertype*$(A,$*Memrel*$(\gamma)))$
⟨*proof*⟩

lemma *cf_idemp*:
assumes *Limit*(γ)
shows $cf(\gamma) = cf(cf(\gamma))$
⟨*proof*⟩

lemma *cf_le_cardinal*:
assumes *Limit*(γ)
shows $cf(\gamma) \leq |\gamma|$
⟨*proof*⟩

lemma *regular_is_Card*:
notes *le_imp_subset* [*dest*]
assumes *Limit*(γ) $\gamma = cf(\gamma)$

shows *Card*(γ)
⟨*proof*⟩

lemma *Limit_cf*: **assumes** *Limit*(κ) **shows** *Limit*(*cf*(κ))
⟨*proof*⟩

lemma *InfCard_cf*: *Limit*(κ) ⟹ *InfCard*(*cf*(κ))
⟨*proof*⟩

lemma *cf_le_cf_fun*:
notes [*dest*] = *Limit_is_Ord*
assumes *cf*(κ) ≤ ν *Limit*(κ)
shows ∃*f*. *f*:ν → κ ∧ *cf_fun*(*f*, κ)
⟨*proof*⟩

lemma *Limit_cofinal_fun_lt*:
notes [*dest*] = *Limit_is_Ord*
assumes *Limit*(κ) *f*: ν → κ *cf_fun*(*f*,κ) *n*∈κ
shows ∃α∈ν. *n* < *f*‘α
⟨*proof*⟩

context
includes *Ord_dests Aleph_dests Aleph_intros Aleph_mem_dests mono_map_rules*
begin

We end this section by calculating the cofinality of Alephs, for the zero and limit case. The successor case depends on AC.

lemma *cf_nat*: *cf*(ω) = ω
⟨*proof*⟩

lemma *cf_Aleph_zero*: $cf(\aleph_0) = \aleph_0$
⟨*proof*⟩

lemma *cf_Aleph_Limit*:
assumes *Limit*(γ)
shows $cf(\aleph_\gamma) = cf(\gamma)$
⟨*proof*⟩

end — includes

end

4 Cardinal Arithmetic under Choice

theory *Cardinal_Library*
imports
ZF_Library
ZF.Cardinal_AC

begin

This theory includes results on cardinalities that depend on AC

4.1 Results on cardinal exponentiation

Non trivial instances of cardinal exponentiation require that the relevant function spaces are well-ordered, hence this implies a strong use of choice.

lemma *cexp_eqpoll_cong*:
assumes
$A \approx A'$ $B \approx B'$
shows
$A^{\uparrow B} = A'^{\uparrow B'}$
⟨*proof*⟩

lemma *cexp_cexp_cmult*: $(\kappa^{\uparrow \nu 1})^{\uparrow \nu 2} = \kappa^{\uparrow \nu 2 \otimes \nu 1}$
⟨*proof*⟩

lemma *cardinal_Pow*: $|Pow(X)| = 2^{\uparrow X}$ — Perhaps it's better with |X|
⟨*proof*⟩

lemma *cantor_cexp*:
assumes $Card(\nu)$
shows $\nu < 2^{\uparrow \nu}$
⟨*proof*⟩

lemma *cexp_left_mono*:
assumes $\kappa 1 \le \kappa 2$
shows $\kappa 1^{\uparrow \nu} \le \kappa 2^{\uparrow \nu}$

⟨*proof*⟩

lemma *cantor_cexp'*:
assumes $2 \le \kappa$ $Card(\nu)$
shows $\nu < \kappa^{\uparrow \nu}$
⟨*proof*⟩

lemma *InfCard_cexp*:
assumes $2 \le \kappa$ $InfCard(\nu)$
shows $InfCard(\kappa^{\uparrow \nu})$
⟨*proof*⟩

lemmas *InfCard_cexp'* = *InfCard_cexp*[*OF nats_le_InfCard*, *simplified*]
— $[\![InfCard(\kappa); InfCard(\nu)]\!] \Longrightarrow InfCard(\kappa^{\uparrow \nu})$

4.2 Miscellaneous

lemma *cardinal_RepFun_le*: $|\{f(a)\ .\ a \in A\}| \le |A|$
⟨*proof*⟩

lemma *subset_imp_le_cardinal*: $A \subseteq B \Longrightarrow |A| \leq |B|$
⟨*proof*⟩

lemma *lt_cardinal_imp_not_subset*: $|A| < |B| \Longrightarrow \neg\ B \subseteq A$
⟨*proof*⟩

lemma *cardinal_lt_csucc_iff*: $Card(K) \Longrightarrow |K'| < K^{+} \longleftrightarrow |K'| \leq K$
⟨*proof*⟩

lemma *cardinal_UN_le_nat*:
$(\bigwedge i.\ i{\in}\omega \Longrightarrow |X(i)| \leq \omega) \Longrightarrow |\bigcup i{\in}\omega.\ X(i)| \leq \omega$
⟨*proof*⟩

lemma *lepoll_imp_cardinal_UN_le*:
notes $[dest] = InfCard\_is\_Card\ Card\_is\_Ord$
assumes $InfCard(K)\ J \lesssim K\ \bigwedge i.\ i{\in}J \Longrightarrow |X(i)| \leq K$
shows $|\bigcup i{\in}J.\ X(i)| \leq K$
⟨*proof*⟩
lemmas *leqpoll_imp_cardinal_UN_le* = *lepoll_imp_cardinal_UN_le*

lemma *cardinal_lt_csucc_iff'*:
includes *Ord_dests*
assumes $Card(\kappa)$
shows $\kappa < |X| \longleftrightarrow \kappa^{+} \leq |X|$
⟨*proof*⟩

lemma *lepoll_imp_subset_bij*: $X \lesssim Y \longleftrightarrow (\exists Z.\ Z \subseteq Y \wedge Z \approx X)$
⟨*proof*⟩

The following result proves to be very useful when combining *cardinal* and $(\approx)$ in a calculation.

lemma *cardinal_Card_eqpoll_iff*: $Card(\kappa) \Longrightarrow |X| = \kappa \longleftrightarrow X \approx \kappa$
⟨*proof*⟩

lemma *lepoll_imp_lepoll_cardinal*: **assumes** $X \lesssim Y$ **shows** $X \lesssim |Y|$
⟨*proof*⟩

lemma *lepoll_Un*:
assumes $InfCard(\kappa)\ A \lesssim \kappa\ B \lesssim \kappa$
shows $A \cup B \lesssim \kappa$
⟨*proof*⟩

lemma *cardinal_Un_le*:
assumes $InfCard(\kappa)\ |A| \leq \kappa\ |B| \leq \kappa$
shows $|A \cup B| \leq \kappa$
⟨*proof*⟩

This is the unconditional version under choice of *Cardinal.Finite_cardinal_iff*.

lemma *Finite_cardinal_iff'*: $Finite(|i|) \longleftrightarrow Finite(i)$
⟨*proof*⟩

lemma *cardinal_subset_of_Card*:
assumes $Card(\gamma)$ $a \subseteq \gamma$
shows $|a| < \gamma \vee |a| = \gamma$
⟨*proof*⟩

lemma *cardinal_cases*:
includes *Ord_dests*
shows $Card(\gamma) \Longrightarrow |X| < \gamma \longleftrightarrow \neg\ |X| \geq \gamma$
⟨*proof*⟩

4.3 Countable and uncountable sets

lemma *countable_iff_cardinal_le_nat*: $countable(X) \longleftrightarrow |X| \leq \omega$
⟨*proof*⟩

lemma *lepoll_countable*: $X \lesssim Y \Longrightarrow countable(Y) \Longrightarrow countable(X)$
⟨*proof*⟩
lemma *surj_countable*: $countable(X) \Longrightarrow f \in surj(X,Y) \Longrightarrow countable(Y)$
⟨*proof*⟩

lemma *Finite_imp_countable*: $Finite(X) \Longrightarrow countable(X)$
⟨*proof*⟩

lemma *countable_imp_countable_UN*:
assumes $countable(J)$ $\bigwedge i.\ i{\in}J \Longrightarrow countable(X(i))$
shows $countable(\bigcup i{\in}J.\ X(i))$
⟨*proof*⟩

lemma *countable_union_countable*:
assumes $\bigwedge x.\ x \in C \Longrightarrow countable(x)$ $countable(C)$
shows $countable(\bigcup C)$
⟨*proof*⟩

abbreviation
uncountable :: $i{\Rightarrow}o$ **where**
$uncountable(X) \equiv \neg\ countable(X)$

lemma *uncountable_iff_nat_lt_cardinal*:
$uncountable(X) \longleftrightarrow \omega < |X|$
⟨*proof*⟩

lemma *uncountable_not_empty*: $uncountable(X) \Longrightarrow X \neq 0$
⟨*proof*⟩

lemma *uncountable_imp_Infinite*: $uncountable(X) \Longrightarrow Infinite(X)$
⟨*proof*⟩

lemma *uncountable_not_subset_countable*:
  assumes $countable(X)$ $uncountable(Y)$
  shows $\neg\ (Y \subseteq X)$
  ⟨*proof*⟩

4.4 Results on Alephs

lemma *nat_lt_Aleph1*: $\omega < \aleph_1$
  ⟨*proof*⟩

lemma *zero_lt_Aleph1*: $0 < \aleph_1$
  ⟨*proof*⟩

lemma *le_aleph1_nat*: $Card(k) \Longrightarrow k<\aleph_1 \Longrightarrow k \leq \omega$
  ⟨*proof*⟩

lemma *Aleph_succ*: $\aleph_{succ(\alpha)} = \aleph_\alpha{}^+$
  ⟨*proof*⟩

lemma *lesspoll_aleph_plus_one*:
  assumes $Ord(\alpha)$
  shows $d \prec \aleph_{succ(\alpha)} \longleftrightarrow d \lesssim \aleph_\alpha$
  ⟨*proof*⟩

lemma *cardinal_Aleph* [*simp*]: $Ord(\alpha) \Longrightarrow |\aleph_\alpha| = \aleph_\alpha$
  ⟨*proof*⟩
lemma *Aleph_lesspoll_increasing*:
  includes *Aleph_intros*
  shows $a < b \Longrightarrow \aleph_a \prec \aleph_b$
  ⟨*proof*⟩

lemma *uncountable_iff_subset_eqpoll_Aleph1*:
  includes *Ord_dests*
  notes *Aleph_zero_eq_nat*[*simp*] *Card_nat*[*simp*] *Aleph_succ*[*simp*]
  shows $uncountable(X) \longleftrightarrow (\exists S.\ S \subseteq X \wedge S \approx \aleph_1)$
⟨*proof*⟩

lemma *lt_Aleph_imp_cardinal_UN_le_nat*: $function(G) \Longrightarrow domain(G) \lesssim \omega$
$\Longrightarrow$
   $\forall n\in domain(G).\ |G`n|<\aleph_1 \Longrightarrow |\bigcup n\in domain(G).\ G`n|\leq\omega$
⟨*proof*⟩

lemma *Aleph1_eq_cardinal_vimage*: $f:\aleph_1\to\omega \Longrightarrow \exists n\in\omega.\ |f\text{-}``\{n\}| = \aleph_1$
⟨*proof*⟩
lemma *eqpoll_Aleph1_cardinal_vimage*:
  assumes $X \approx \aleph_1$ $f : X \to \omega$
  shows $\exists n\in\omega.\ |f\text{-}``\{n\}| = \aleph_1$
⟨*proof*⟩

4.5 Applications of transfinite recursive constructions

The next lemma is an application of recursive constructions. It works under the assumption that whenever the already constructed subsequence is small enough, another element can be added.

lemma *bounded_cardinal_selection*:
  includes *Ord_dests*
  assumes
    $\bigwedge X.\ |X| < \gamma \Longrightarrow X \subseteq G \Longrightarrow \exists\, a{\in}G.\ \forall\, s{\in}X.\ Q(s,a)$ $b{\in}G$ $Card(\gamma)$
  shows
    $\exists\, S.\ S : \gamma \to G \wedge (\forall\, \alpha \in \gamma.\ \forall\, \beta \in \gamma.\ \ \alpha{<}\beta \longrightarrow Q(S`\alpha, S`\beta))$
⟨*proof*⟩

The following basic result can, in turn, be proved by a bounded-cardinal selection.

lemma *Infinite_iff_lepoll_nat*: $Infinite(X) \longleftrightarrow \omega \lesssim X$
⟨*proof*⟩

lemma *Infinite_InfCard_cardinal*: $Infinite(X) \Longrightarrow InfCard(|X|)$
  ⟨*proof*⟩

lemma *Finite_to_one_surj_imp_cardinal_eq*:
  assumes $F \in Finite\_to\_one(X,Y) \cap surj(X,Y)$ $Infinite(X)$
  shows $|Y| = |X|$
⟨*proof*⟩

lemma *cardinal_map_Un*:
  assumes $Infinite(X)$ $Finite(b)$
  shows $|\{a \cup b\ .\ a \in X\}| = |X|$
⟨*proof*⟩

end
theory *Konig*
  imports
    Cofinality
    Cardinal_Library

begin

Now, using the Axiom of choice, we can show that all successor cardinals are regular.

lemma *cf_csucc*:
  assumes $InfCard(z)$
  shows $cf(z^+) = z^+$
⟨*proof*⟩

And this finishes the calculation of cofinality of Alephs.

lemma *cf_Aleph_succ*: $Ord(z) \Longrightarrow cf(\aleph_{succ(z)}) = \aleph_{succ(z)}$

⟨*proof*⟩

4.6 König's Theorem

We end this section by proving König's Theorem on the cofinality of cardinal exponentiation. This is a strengthening of Cantor's theorem and it is essentially the only basic way to prove strict cardinal inequalities.

It is proved rather straightforwardly with the tools already developed.

lemma *konigs_theorem*:
  notes [*dest*] = *InfCard_is_Card Card_is_Ord*
    and [*trans*] = *lt_trans1 lt_trans2*
  assumes
    $InfCard(\kappa)$ $InfCard(\nu)$ $cf(\kappa) \le \nu$
  shows
    $\kappa < \kappa^{\uparrow\nu}$
  ⟨*proof*⟩

lemma *cf_cexp*:
  assumes
    $Card(\kappa)$ $InfCard(\nu)$ $2 \le \kappa$
  shows
    $\nu < cf(\kappa^{\uparrow\nu})$
⟨*proof*⟩

Finally, the next two corollaries illustrate the only possible exceptions to the value of the cardinality of the continuum: The limit cardinals of countable cofinality. That these are the only exceptions is a consequence of Easton's Theorem [4, Thm 15.18].

corollary *cf_continuum*: $\aleph_0 < cf(2^{\uparrow\aleph_0})$
  ⟨*proof*⟩

corollary *continuum_not_eq_Aleph_nat*: $2^{\uparrow\aleph_0} \neq \aleph_\omega$
  ⟨*proof*⟩

end

5 The Delta System Lemma

theory *Delta_System*
  imports
    Cardinal_Library

begin

The *Delta System Lemma* (DSL) states that any uncountable family of finite sets includes an uncountable delta system. This is the simplest non trivial version; others, for cardinals greater than $\aleph_1$ assume some weak versions of the generalized continuum hypothesis for the cardinals involved.

The proof is essentially the one in [6, III.2.6] for the case $\aleph_1$; another similar presentation can be found in [5, Chap. 16].

lemma *delta_system_Aleph1*:
  assumes $\forall A{\in}F.\ Finite(A)$ $F \approx \aleph_1$
  shows $\exists D.\ D \subseteq F \wedge delta\_system(D) \wedge D \approx \aleph_1$
⟨*proof*⟩

lemma *delta_system_uncountable*:
  assumes $\forall A{\in}F.\ Finite(A)$ $uncountable(F)$
  shows $\exists D.\ D \subseteq F \wedge delta\_system(D) \wedge D \approx \aleph_1$
⟨*proof*⟩

end

5.1 Application to Cohen posets

theory *Cohen_Posets*
  imports
    Delta_System

begin

We end this session by applying DSL to the combinatorics of finite function posets. We first define some basic concepts; we take a different approach from [1], in that the order relation is presented as a predicate (of type $i \Rightarrow i \Rightarrow o$).

Two elements of a poset are *compatible* if they have a common lower bound.

definition *compat_in* :: $[i,[i,i]\Rightarrow o,i,i]\Rightarrow o$ **where**
  $compat\_in(A,r,p,q) \equiv \exists d{\in}A\ .\ r(d,p) \wedge r(d,q)$

An *antichain* is a subset of pairwise incompatible members.

definition
  antichain :: $[i,[i,i]\Rightarrow o,i]\Rightarrow o$ **where**
  $antichain(P,leq,A) \equiv A{\subseteq}P \wedge (\forall p{\in}A.\ \forall q{\in}A.$
      $p{\neq}q \longrightarrow \neg compat\_in(P,leq,p,q))$

A poset has the *countable chain condition* (ccc) if all of its antichains are countable.

definition
  ccc :: $[i,[i,i]\Rightarrow o]\Rightarrow o$ **where**
  $ccc(P,leq) \equiv \forall A.\ antichain(P,leq,A) \longrightarrow countable(A)$

Finally, the *Cohen poset* is the set of finite partial functions between two sets with the order of reverse inclusion.

definition

Fn :: $[i,i] \Rightarrow i$ **where**
$Fn(I,J) \equiv \bigcup\{(d{\to}J) \ . \ d \in \{x \in Pow(I). \ \ Finite(x)\}\}$

abbreviation
$Supset$ :: $i \Rightarrow i \Rightarrow o$ (**infixl** ‹$\supseteq$› 50) **where**
$f \supseteq g \equiv g \subseteq f$

lemma $FnI[intro]$:
assumes $p : d \to J$ $d \subseteq I$ $Finite(d)$
shows $p \in Fn(I,J)$
⟨*proof*⟩

lemma $FnD[dest]$:
assumes $p \in Fn(I,J)$
shows $\exists d. \ p : d \to J \wedge d \subseteq I \wedge Finite(d)$
⟨*proof*⟩

lemma *Fn_is_function*: $p \in Fn(I,J) \Longrightarrow function(p)$
⟨*proof*⟩

lemma *restrict_eq_imp_compat*:
assumes $f \in Fn(I, \ J)$ $g \in Fn(I, \ J)$
$restrict(f, \ domain(f) \cap domain(g)) = restrict(g, \ domain(f) \cap domain(g))$
shows $f \cup g \in Fn(I, \ J)$
⟨*proof*⟩

We finally arrive to our application of DSL.

lemma *ccc_Fn_2*: $ccc(Fn(I,2), \ (\supseteq))$
⟨*proof*⟩

The fact that a poset P has the ccc has useful consequences for the theory of forcing, since it implies that cardinals from the original model are exactly the cardinals in any generic extension by P [6, Chap. IV].

end

References

[1] E. Gunther, M. Pagano, P. Sánchez Terraf, Formalization of Forcing in Isabelle/ZF, in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Automated Reasoning. 10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part II, Lecture Notes in Artificial Intelligence **12167**, Springer International Publishing: 221–235 (2020). doi:10.1007/978-3-030-51054-1.

[2] E. Gunther, M. Pagano, P. Sánchez Terraf, M. Steinberg, The independence of the continuum hypothesis in Isabelle/ZF, *Archive of Formal Proofs* (2022). https://isa-afp.org/entries/Independence_CH.html, Formal proof development.

[3] E. Gunther, M. Pagano, P. Sánchez Terraf, M. Steinberg, Transitive models of fragments of ZFC, *Archive of Formal Proofs* (2022). https://isa-afp.org/entries/Transitive_Models.html, Formal proof development.

[4] T. Jech, "Set Theory. The Millennium Edition", Springer Monographs in Mathematics, Springer-Verlag (2002), third edition. Corrected fourth printing, 2006.

[5] W. Just, M. Weese, "Discovering Modern Set Theory. II", Grad. Studies in Mathematics **18**, American Mathematical Society (1997).

[6] K. Kunen, "Set Theory", Studies in Logic, College Publications (2011), second edition. Revised edition, 2013.

[7] L.C. Paulson, K. Grabczewski, Mechanizing set theory, *J. Autom. Reasoning* **17**: 291–323 (1996). doi:10.1007/BF00283132.

[8] P. Sánchez Terraf, Course notes on set theory, online, (2019). In Spanish, https://cs.famaf.unc.edu.ar/~pedro/home_en.html#set_theory.