Wetzel's Problem and the Continuum Hypothesis

Lawrence C. Paulson

March 17, 2025

Abstract

Let F be a set of analytic functions on the complex plane such that, for each $z \in \mathbb{C}$, the set $\{f(z) \mid f \in F\}$ is countable; must then F itself be countable? The answer is yes if the Continuum Hypothesis is false, i.e., if the cardinality of $\mathbb{R}$ exceeds $\aleph_1$. But if CH is true then such an F, of cardinality $\aleph_1$, can be constructed by transfinite recursion.

The formal proof illustrates reasoning about complex analysis (analytic and homomorphic functions) and set theory (transfinite cardinalities) in a single setting. The mathematical text comes from *Proofs from THE BOOK* [1, pp. 137–8], by Aigner and Ziegler.

Contents

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council. Thanks also to Manuel Eberl for advice on proving a function to be holomorphic.

1 Wetzel's Problem, Solved by Erdös

Martin Aigner and Günter M. Ziegler. Proofs from THE BOOK. (Springer, 2018). Chapter 19: Sets, functions, and the continuum hypothesis Theorem 5 (pages 137–8)

theory *Wetzels-Problem* **imports**
HOL−Complex-Analysis.Complex-Analysis ZFC-in-HOL.General-Cardinals

begin

definition *Wetzel* :: (*complex* ⇒ *complex*) *set* ⇒ *bool*
where *Wetzel* ≡ λ*F*. (∀ *f*∈*F*. *f analytic-on UNIV*) ∧ (∀ *z*. *countable*((λ*f*. *f z*) ' *F*))

1.0.1 When the continuum hypothesis is false

proposition *Erdos-Wetzel-nonCH*:
assumes *W*: *Wetzel F* **and** *NCH*: *C-continuum* > ℵ*1*
shows *countable F*
proof −
have ∃ *z0*. *gcard* ((λ*f*. *f z0*) ' *F*) ≥ ℵ*1* **if** *uncountable F*
proof −
have *gcard F* ≥ ℵ*1*
using *that uncountable-gcard-ge* **by** *force*
then obtain *F′* **where** *F′* ⊆ *F* **and** *F′*: *gcard F′* = ℵ*1*
by (*meson Card-Aleph subset-smaller-gcard*)
then obtain φ **where** φ: *bij-betw* φ (*elts* ω*1*) *F′*
by (*metis TC-small eqpoll-def gcard-eqpoll*)
define *S* **where** *S* ≡ λα β. {*z*. φ α *z* = φ β *z*}
have *co-S*: *gcard* (*S* α β) ≤ ℵ*0* **if** α ∈ *elts* β β ∈ *elts* ω*1* **for** α β
proof −
have φ α *holomorphic-on UNIV* φ β *holomorphic-on UNIV*
using *W* ‹*F′* ⊆ *F*› **unfolding** *Wetzel-def*
by (*meson Ord-*ω*1 Ord-trans* φ *analytic-imp-holomorphic bij-betwE subsetD that*)+
moreover have φ α ≠ φ β
by (*metis Ord-*ω*1 Ord-trans* φ *bij-betw-def inj-on-def mem-not-refl that*)
ultimately have *countable* (*S* α β)
using *holomorphic-countable-equal-UNIV* **unfolding** *S-def* **by** *blast*
then show *?thesis*
using *countable-imp-g-le-Aleph0* **by** *blast*
qed
define *SS* **where** *SS* ≡⨆β ∈ *elts* ω*1*. ⨆α ∈ *elts* β. *S* α β
have *F′-eq*: *F′* = φ ' *elts* ω*1*
using φ *bij-betw-imp-surj-on* **by** *auto*
have §: ⋀β. β ∈ *elts* ω*1* ⟹ *gcard* (⋃α∈*elts* β. *S* α β) ≤ ω
by (*metis Aleph-0 TC-small co-S countable-UN countable-iff-g-le-Aleph0 less-*ω*1-imp-countable*)
have *gcard SS* ≤ *gcard* ((λβ. ⋃α∈*elts* β. *S* α β) ' *elts* ω*1*) ⊗ ℵ*0*

```
    apply (simp add: SS-def)
    by (metis (no-types, lifting) § TC-small gcard-Union-le-cmult imageE)
  also have ... ≤ ℵ1
  proof (rule cmult-InfCard-le)
    show gcard ((λβ. ⋃α∈elts β. S α β) ` elts ω1) ≤ ω1
      using gcard-image-le by fastforce
  qed auto
  finally have gcard SS ≤ ℵ1 .
  with NCH obtain z0 where z0 ∉ SS
    by (metis Complex-gcard UNIV-eq-I less-le-not-le)
  then have inj-on (λx. φ x z0) (elts ω1)
    apply (simp add: SS-def S-def inj-on-def)
    by (metis Ord-ω1 Ord-in-Ord Ord-linear)
  then have gcard ((λf. f z0) ` F') = ℵ1
    by (smt (verit) F' F'-eq gcard-image imageE inj-on-def)
  then show ?thesis
    by (metis TC-small ‹F' ⊆ F› image-mono subset-imp-gcard-le)
 qed
 with W show ?thesis
  unfolding Wetzel-def by (meson countable uncountable-gcard-ge)
qed
```

1.0.2 When the continuum hypothesis is true

```
lemma Rats-closure-real2: closure (ℚ×ℚ) = (UNIV::real set)×(UNIV::real set)
 by (simp add: Rats-closure-real closure-Times)

proposition Erdos-Wetzel-CH:
 assumes CH: C-continuum = ℵ1
 obtains F where Wetzel F and uncountable F
proof −
 define D where D ≡ {z. Re z ∈ ℚ ∧ Im z ∈ ℚ}
 have Deq: D = (⋃x∈ℚ. ⋃y∈ℚ. {Complex x y})
  using complex.collapse by (force simp: D-def)
 with countable-rat have countable D
  by blast
 have infinite D
  unfolding Deq
  by (intro infinite-disjoint-family-imp-infinite-UNION Rats-infinite) (auto simp:
disjoint-family-on-def)
 have ∃w. Re w ∈ ℚ ∧ Im w ∈ ℚ ∧ norm (w − z) < e if e > 0 for z and e::real
 proof −
  obtain x y where x∈ℚ y∈ℚ and xy: dist (x,y) (Re z, Im z) < e
    using ‹e > 0› Rats-closure-real2 unfolding closure-approachable set-eq-iff
    by blast
  moreover have dist (x,y) (Re z, Im z) = norm (Complex x y − z)
    by (simp add: norm-complex-def norm-prod-def dist-norm)
  ultimately show ∃w. Re w ∈ ℚ ∧ Im w ∈ ℚ ∧ norm (w − z) < e
    by (metis complex.sel)
```

qed
then have *cloD*: *closure D* = *UNIV*
by (*auto simp*: *D-def closure-approachable dist-complex-def*)
obtain ζ **where** ζ: *bij-betw* ζ (*elts* $\omega 1$) (*UNIV*::*complex set*)
by (*metis Complex-gcard TC-small assms eqpoll-def gcard-eqpoll*)
define *inD* **where** *inD* $\equiv \lambda\beta\ f.\ (\forall\alpha \in elts\ \beta.\ f\ (\zeta\ \alpha) \in D)$
define Φ **where** $\Phi \equiv \lambda\beta\ f.\ f\ \beta$ *analytic-on UNIV* $\wedge$ *inD* $\beta\ (f\ \beta) \wedge$ *inj-on f* (*elts* (*succ* β))
have *ind-step*: $\exists h.\ \Phi\ \gamma$ ((*restrict f* (*elts* γ))(γ:=h))
if γ: $\gamma \in$ *elts* $\omega 1$ **and** $\forall\beta \in elts\ \gamma.\ \Phi\ \beta\ f$ **for** $\gamma\ f$
proof −
have *f*: $\forall\beta \in elts\ \gamma.\ f\ \beta$ *analytic-on UNIV* $\wedge$ *inD* $\beta\ (f\ \beta)$
using *that* **by** (*auto simp*: Φ*-def*)
have *inj*: *inj-on f* (*elts* γ)
using *that* **by** (*simp add*: Φ*-def inj-on-def*) (*meson Ord-*$\omega 1$ *Ord-in-Ord Ord-linear*)
obtain *h* **where** *h analytic-on UNIV inD* γ *h* $(\forall\beta \in elts\ \gamma.\ h \neq f\ \beta)$
proof (*cases finite* (*elts* γ))
case *True*
then obtain η **where** η: *bij-betw* η {..<*card* (*elts* γ)} (*elts* γ)
using *bij-betw-from-nat-into-finite* **by** *blast*
define *g* **where** $g \equiv f\ o\ \eta$
define *w* **where** $w \equiv \zeta\ o\ \eta$
have *gf*: $\forall i<card\ (elts\ \gamma).\ h \neq g\ i \Longrightarrow \forall\beta \in elts\ \gamma.\ h \neq f\ \beta$ **for** *h*
using η **by** (*auto simp*: *bij-betw-iff-bijections g-def*)
have ∗∗: $\exists h.\ h$ *analytic-on UNIV* $\wedge$ $(\forall i<n.\ h\ (w\ i) \in D \wedge h\ (w\ i) \neq g\ i\ (w\ i))$
if $n \leq card\ (elts\ \gamma)$ **for** *n*
using *that*
proof (*induction n*)
case *0*
then show *?case*
using *analytic-on-const* **by** *blast*
next
case (*Suc n*)
then obtain *h* **where** *h analytic-on UNIV* **and** *hg*: $\forall i<n.\ h(w\ i) \in D \wedge h(w\ i) \neq g\ i\ (w\ i)$
using *Suc-leD* **by** *blast*
define *p* **where** $p \equiv \lambda z.\ \prod i<n.\ z - w\ i$
have *p0*: $p\ z = 0 \longleftrightarrow (\exists i<n.\ z = w\ i)$ **for** *z*
unfolding *p-def* **by** *force*
obtain *d* **where** *d*: $d \in D - \{g\ n\ (w\ n)\}$
using ‹*infinite D*› **by** (*metis ex-in-conv finite.emptyI infinite-remove*)
define *h'* **where** $h' \equiv \lambda z.\ h\ z + p\ z * (d - h\ (w\ n))\ /\ p\ (w\ n)$
have *h'-eq*: $h'\ (w\ i) = h\ (w\ i)$ **if** $i<n$ **for** *i*
using *that* **by** (*force simp*: *h'-def p0*)
show *?case*
proof (*intro exI strip conjI*)
have *nless*: $n < card\ (elts\ \gamma)$

using *Suc.prems Suc-le-eq* **by** *blast*
with η **have** $\eta\ n \neq \eta\ i$ **if** $i<n$ **for** i
using *that* **unfolding** *bij-betw-iff-bijections*
by (*metis lessThan-iff less-not-refl order-less-trans*)
with $\zeta\ \eta\ \gamma$ **have** *pwn-nonzero*: $p\ (w\ n) \neq 0$
apply (*clarsimp simp*: *p0 w-def bij-betw-iff-bijections*)
by (*metis Ord-ω1 Ord-trans nless lessThan-iff order-less-trans*)
then show h' *analytic-on UNIV*
unfolding h'*-def p-def* **by** (*intro analytic-intros* ‹*h analytic-on UNIV*›)
fix i
assume $i < Suc\ n$
then have §: $i < n \lor i = n$
by *linarith*
then show $h'\ (w\ i) \in D$
using h'*-eq hg d* h'*-def pwn-nonzero* **by** *force*
show $h'\ (w\ i) \neq g\ i\ (w\ i)$
using § h'*-eq hg* h'*-def d pwn-nonzero* **by** *fastforce*
qed
qed
show *?thesis*
using $**$ [*OF order-refl*] η *that gf*
by (*simp add*: *w-def bij-betw-iff-bijections inD-def*) *metis*
next
case *False*
then obtain η **where** η: *bij-betw* η (*UNIV*::*nat set*) (*elts* γ)
by (*meson* γ *countable-infiniteE′ less-ω1-imp-countable*)
then have η*-inject* [*simp*]: $\eta\ i = \eta\ j \longleftrightarrow i=j$ **for** $i\ j$
by (*simp add*: *bij-betw-imp-inj-on inj-eq*)
define g **where** $g \equiv f\ o\ \eta$
define w **where** $w \equiv \zeta\ o\ \eta$
then have *w-inject* [*simp*]: $w\ i = w\ j \longleftrightarrow i=j$ **for** $i\ j$
by (*smt* (*verit*) *Ord-ω1 Ord-trans UNIV-I* $\eta\ \gamma\ \zeta$ *bij-betw-iff-bijections comp-apply*)
define p **where** $p \equiv \lambda n\ z.\ \prod i<n.\ z - w\ i$
define q **where** $q \equiv \lambda n.\ \prod i<n.\ 1 + norm\ (w\ i)$
define h **where** $h \equiv \lambda n\ \varepsilon\ z.\ \sum i<n.\ \varepsilon\ i * p\ i\ z$
define *BALL* **where** $BALL \equiv \lambda n\ \varepsilon.\ ball\ (h\ n\ \varepsilon\ (w\ n))\ (norm\ (p\ n\ (w\ n))\ /\ (fact\ n * q\ n))$
— The demonimator above is the key to keeping the epsilons small
define *DD* **where** $DD \equiv \lambda n\ \varepsilon.\ D \cap BALL\ n\ \varepsilon - \{g\ n\ (w\ n)\}$
define *dd* **where** $dd \equiv \lambda n\ \varepsilon.\ SOME\ x.\ x \in DD\ n\ \varepsilon$
have *p0*: $p\ n\ z = 0 \longleftrightarrow (\exists i<n.\ z = w\ i)$ **for** $z\ n$
unfolding *p-def* **by** *force*
have [*simp*]: $p\ n\ (w\ i) = 0$ **if** $i<n$ **for** $i\ n$
using *that* **by** (*simp add*: *p0*)
have *q-gt0*: $0 < q\ n$ **for** n
unfolding *q-def* **by** (*smt* (*verit*) *norm-not-less-zero prod-pos*)
have $DD\ n\ \varepsilon \neq \{\}$ **for** $n\ \varepsilon$
proof −

have $r > 0 \Longrightarrow$ *infinite* ($D \cap$ *ball z r*) **for** *z r*
by (*metis islimpt-UNIV limpt-of-closure islimpt-eq-infinite-ball cloD*)
then have *infinite* ($D \cap$ *BALL n* ε) **for** *n* ε
by (*simp add*: *BALL-def p0 q-gt0*)
then show *?thesis*
by (*metis DD-def finite.emptyI infinite-remove*)
qed
then have *dd-in-DD*: *dd n* $\varepsilon \in$ *DD n* ε **for** *n* ε
by (*simp add*: *dd-def some-in-eq*)

have *h-cong*: *h n* ε = *h n* ε' **if** $\bigwedge i.\ i<n \Longrightarrow \varepsilon\ i = \varepsilon'\ i$ **for** *n* ε ε'
using *that* **by** (*simp add*: *h-def*)
have *dd-cong*: *dd n* ε = *dd n* ε' **if** $\bigwedge i.\ i<n \Longrightarrow \varepsilon\ i = \varepsilon'\ i$ **for** *n* ε ε'
using *that* **by** (*metis dd-def DD-def BALL-def h-cong*)
have [*simp*]: *h n* (*cut* ε *less-than n*) = *h n* ε **for** *n* ε
by (*meson cut-apply h-cong less-than-iff*)
have [*simp*]: *dd n* (*cut* ε *less-than n*) = *dd n* ε **for** *n* ε
by (*meson cut-apply dd-cong less-than-iff*)

define *coeff* **where** *coeff* $\equiv$ *wfrec less-than* ($\lambda\varepsilon$ *n*. (*dd n* ε − *h n* ε (*w n*)) / *p n* (*w n*))
have *coeff-eq*: *coeff n* = (*dd n coeff* − *h n coeff* (*w n*)) / *p n* (*w n*) **for** *n*
by (*simp add*: *def-wfrec* [*OF coeff-def*])

have *norm-coeff*: *norm* (*coeff n*) < *1* / (*fact n* ∗ *q n*) **for** *n*
using *dd-in-DD* [*of n coeff*]
by (*simp add*: *q-gt0 coeff-eq DD-def BALL-def dist-norm norm-minus-commute norm-divide divide-simps*)
have *norm-p-bound*: *norm* (*p n z'*) ≤ *q n* ∗ (*1* + *norm z*) ^ *n*
if *dist z z'* ≤ *1* **for** *n z z'*
proof (*induction n*)
case *0*
then show *?case*
by (*auto simp*: *p-def q-def*)
next
case (*Suc n*)
have *norm z'* − *norm z* ≤ *1*
by (*smt* (*verit*) *dist-norm norm-triangle-ineq3 that*)
then have §: *norm* (*z'* − *w n*) ≤ (*1* + *norm* (*w n*)) ∗ (*1* + *norm z*)
by (*simp add*: *mult.commute add-mono distrib-left norm-triangle-le-diff*)
have *norm* (*p n z'*) ∗ *norm* (*z'* − *w n*) ≤ (*q n* ∗ (*1* + *norm z*) ^ *n*) ∗ *norm* (*z'* − *w n*)
by (*metis Suc mult.commute mult-left-mono norm-ge-zero*)
also have ... ≤ (*q n* ∗ (*1* + *norm z*) ^ *n*) ∗ (*1* + *norm* (*w n*)) ∗ ((*1* + *norm z*))
by (*smt* (*verit*) § *Suc mult.assoc mult-left-mono norm-ge-zero*)
also have ... ≤ *q n* ∗ (*1* + *norm* (*w n*)) ∗ ((*1* + *norm z*) ∗ (*1* + *norm z*) ^ *n*)
by *auto*

finally show *?case*
by (*auto simp*: *p-def q-def norm-mult simp del*: *fact-Suc*)
qed

show *?thesis*
proof
define *hh* **where** *hh* ≡ λ*z*. *suminf* (λ*i*. *coeff i* ∗ *p i z*)
have *hh holomorphic-on UNIV*
proof (*rule holomorphic-uniform-sequence*)
fix *n* — Many thanks to Manuel Eberl for suggesting these approach
show *h n coeff holomorphic-on UNIV*
unfolding *h-def p-def* **by** (*intro holomorphic-intros*)
next
fix *z*
have *uniform-limit* (*cball z 1*) (λ*n*. *h n coeff*) *hh sequentially*
unfolding *hh-def h-def*
proof (*rule Weierstrass-m-test*)
let *?M* = λ*n*. (*1* + *norm z*) ^ *n* / *fact n*
have ∃ *N*. ∀ *n*≥*N*. *B* ≤ (*1* + *real n*) / (*1* + *norm z*) **for** *B*
proof
show ∀ *n*≥*nat* ⌈*B* ∗ (*1* + *norm z*)⌉. *B* ≤ (*1* + *real n*) / (*1* + *norm z*)
using *norm-ge-zero* [*of z*] **by** (*auto simp*: *divide-simps simp del*: *norm-ge-zero*)
qed
then have *L*: *liminf* (λ*n*. *ereal* ((*1* + *real n*) / (*1* + *norm z*))) = ∞
by (*simp add*: *Lim-PInfty flip*: *liminf-PInfty*)
have $\forall_F$ *n in sequentially*. *0* < (*1* + *cmod z*) ^ *n* / *fact n*
using *norm-ge-zero* [*of z*] **by** (*simp del*: *norm-ge-zero*)
then show *summable ?M*
by (*rule ratio-test-convergence*) (*auto simp*: *add-nonneg-eq-0-iff L*)
fix *n z'*
assume *z'* ∈ *cball z 1*
then have *norm* (*coeff n* ∗ *p n z'*) ≤ *norm* (*coeff n*) ∗ *q n* ∗ (*1* + *norm z*) ^ *n*
by (*simp add*: *mult.assoc mult-mono norm-mult norm-p-bound*)
also have . . . ≤ (*1* / *fact n*) ∗ (*1* + *norm z*) ^ *n*
proof (*rule mult-right-mono*)
show *norm* (*coeff n*) ∗ *q n* ≤ *1* / *fact n*
using *q-gt0 norm-coeff* [*of n*] **by** (*simp add*: *field-simps*)
qed *auto*
also have . . . ≤ *?M n*
by (*simp add*: *divide-simps*)
finally show *norm* (*coeff n* ∗ *p n z'*) ≤ *?M n* **.**
qed
then show ∃ *d*>*0*. *cball z d* ⊆ *UNIV* ∧ *uniform-limit* (*cball z d*) (λ*n*. *h n coeff*) *hh sequentially*
using *zero-less-one* **by** *blast*
qed *auto*
then show *hh analytic-on UNIV*

by (*simp add*: *analytic-on-open*)
have *hh-eq-dd*: *hh* (*w n*) = *dd n coeff* **for** *n*
proof −
have *hh* (*w n*) = *h* (*Suc n*) *coeff* (*w n*)
unfolding *hh-def h-def* **by** (*intro suminf-finite*) *auto*
also have . . . = *dd n coeff*
by (*induction n*) (*auto simp add*: *p0 h-def p-def coeff-eq* [*of Suc* -] *coeff-eq* [*of 0*])
finally show *?thesis* **.**
qed
then have *hh* (*w n*) ∈ *D* **for** *n*
using *DD-def dd-in-DD* **by** *fastforce*
then show *inD* γ *hh*
unfolding *inD-def* **by** (*metis* η *bij-betw-iff-bijections comp-apply w-def*)
have *hh* (*w n*) ≠ *f* (η *n*) (*w n*) **for** *n*
using *DD-def dd-in-DD g-def hh-eq-dd* **by** *auto*
then show ∀β∈*elts* γ. *hh* ≠ *f* β
by (*metis* η *bij-betw-imp-surj-on imageE*)
qed
qed
with *f* **show** *?thesis*
using *inj* **by** (*rule-tac x=h* **in** *exI*) (*auto simp*: Φ-*def inj-on-def*)
qed
define *G* **where** *G* ≡ λ*f* γ. @*h*. Φ γ ((*restrict f* (*elts* γ))(γ:=*h*))
define *f* **where** *f* ≡ *transrec G*
have Φ*f*: Φ β *f* **if** β ∈ *elts* ω*1* **for** β
using *that*
proof (*induction* β *rule*: *eps-induct*)
case (*step* γ)
then have *IH*: ∀β∈*elts* γ. Φ β *f*
using *Ord-*ω*1 Ord-trans* **by** *blast*
have *f* γ = *G f* γ
by (*metis G-def f-def restrict-apply′ restrict-ext transrec*)
moreover have Φ γ ((*restrict f* (*elts* γ))(γ := *G f* γ))
by (*metis ind-step*[*OF step.prems*] *G-def IH someI*)
ultimately show *?case*
by (*metis IH* Φ-*def elts-succ fun-upd-same fun-upd-triv inj-on-restrict-eq restrict-upd*)
qed
then have *anf*: ⋀β. β ∈ *elts* ω*1* ⟹ *f* β *analytic-on UNIV*
and *inD*: ⋀α β. ⟦β ∈ *elts* ω*1*; α ∈ *elts* β⟧ ⟹ *f* β (ζ α) ∈ *D*
using Φ-*def inD-def* **by** *blast+*
have *injf*: *inj-on f* (*elts* ω*1*)
using Φ*f* **unfolding** *inj-on-def* Φ-*def* **by** (*metis Ord-*ω*1 Ord-in-Ord Ord-linear-le in-succ-iff*)
show *?thesis*
proof
let *?F* = *f* ‘ *elts* ω*1*
have *countable* ((λ*f*. *f z*) ‘ *f* ‘ *elts* ω*1*) **for** *z*

```
    proof −
      obtain α where α: ζ α = z α ∈ elts ω1 Ord α
        by (meson Ord-ω1 Ord-in-Ord UNIV-I ζ bij-betw-iff-bijections)
      let ?B = elts ω1 − elts (succ α)
      have eq: elts ω1 = elts (succ α) ∪ ?B
       using α by (metis Diff-partition Ord-ω1 OrdmemD less-eq-V-def succ-le-iff)
      have (λf. f z) ‘ f ‘ ?B ⊆ D
        using α inD by clarsimp (meson Ord-ω1 Ord-in-Ord Ord-linear)
      then have countable ((λf. f z) ‘ f ‘ ?B)
        by (meson ‹countable D› countable-subset)
      moreover have countable ((λf. f z) ‘ f ‘ elts (succ α))
        by (simp add: α less-ω1-imp-countable)
      ultimately show ?thesis
        using eq by (metis countable-Un-iff image-Un)
    qed
    then show Wetzel ?F
      unfolding Wetzel-def by (blast intro: anf)
    show uncountable ?F
      using Ord-ω1 countable-iff-less-ω1 countable-image-inj-eq injf by blast
  qed
qed

theorem Erdos-Wetzel: C-continuum = ℵ1 ⟷ (∃F. Wetzel F ∧ uncountable F)
  by (metis C-continuum-ge Erdos-Wetzel-CH Erdos-Wetzel-nonCH less-V-def)
```

The originally submitted version of this theory included the development of cardinals for general Isabelle/HOL sets (as opposed to ZF sets, elements of type V), as well as other generally useful library material. From March 2022, that material has been moved to the analysis libraries or to *ZFC-in-HOL.General-Cardinals*, as appropriate.

end

References

[1] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 6th edition, 2018.