Zeckendorf's Theorem

Christian Dalvit

June 12, 2023

Abstract

This work formalizes Zeckendorf's theorem. The theorem states that every positive integer can be uniquely represented as a sum of one or more non-consecutive Fibonacci numbers. More precisely, if N is a positive integer, there exist positive integers $c_i \geq 2$ with $c_{i+1} > c_i + 1$, such that

$$N = \sum_{i=0}^{k} F_{c_i}$$

where F_n is the n-th Fibonacci number. This entry formalizes the proof from Gerrit Lekkerkerker's paper [1].

Contents

1 Zeckendorf's Theorem

theory *Zeckendorf*

imports
Main
HOL−Number-Theory.Number-Theory

begin

1.1 Definitions

Formulate auxiliary definitions. An increasing sequence is a predicate of a function f together with a set I. f is an increasing sequence on I, if $f(x) + 1 < f(x + 1)$ for all $x \in I$. This definition is used to ensure that the Fibonacci numbers in the sum are non-consecutive.

definition *is-fib* :: *nat* ⇒ *bool* **where**
is-fib $n = (\exists\ i.\ n = fib\ i)$

definition *le-fib-idx-set* :: *nat* ⇒ *nat set* **where**
le-fib-idx-set $n = \{i\ .fib\ i < n\}$

definition *inc-seq-on* :: (*nat* ⇒ *nat*) ⇒ *nat set* ⇒ *bool* **where**
inc-seq-on $f\ I = (\forall\ n \in I.\ f(Suc\ n) > Suc(f\ n))$

definition *fib-idx-set* :: *nat* ⇒ *nat set* **where**
fib-idx-set $n = \{i.\ fib\ i = n\}$

1.2 Auxiliary Lemmas

lemma *fib-values*[*simp*]:
$fib\ 3 = 2$
$fib\ 4 = 3$
$fib\ 5 = 5$
$fib\ 6 = 8$
by(*auto simp*: *numeral-Bit0 numeral-eq-Suc*)

lemma *fib-strict-mono*: $i \geq 2 \Longrightarrow fib\ i < fib\ (Suc\ i)$
using *fib-mono* **by**(*induct i*, *simp*, *fastforce*)

lemma *smaller-index-implies-fib-le*: $i < j \Longrightarrow fib(Suc\ i) \leq fib\ j$
using *fib-mono* **by** (*induct j*, *auto*)

lemma *fib-index-strict-mono* : $i \geq 2 \Longrightarrow j > i \Longrightarrow fib\ j > fib\ i$
by(*induct i*, *simp*, *metis Suc-leI fib-mono fib-strict-mono nle-le nless-le*)

lemma *fib-implies-is-fib*: $fib\ i = n \Longrightarrow$ *is-fib* n
using *is-fib-def* **by** *auto*

lemma *zero-fib-unique-idx*: $n = fib\ i \Longrightarrow n = fib\ 0 \Longrightarrow i = 0$
using *fib-neq-0-nat fib-idx-set-def* **by** *fastforce*

lemma *zero-fib-equiv*: $fib\ i = 0 \longleftrightarrow i = 0$
using *zero-fib-unique-idx* **by** *auto*

lemma *one-fib-idxs*: $fib\ i = Suc\ 0 \Longrightarrow i = Suc\ 0 \lor i = Suc(Suc\ 0)$
using *Fib.fib0 One-nat-def Suc-1 eq-imp-le fib-2 fib-index-strict-mono less-2-cases nat-neq-iff* **by** *metis*

lemma *ge-two-eq-fib-implies-eq-idx*: $n \geq 2 \Longrightarrow n = fib\ i \Longrightarrow n = fib\ j \Longrightarrow i = j$
using *fib-index-strict-mono fib-mono Suc-1 fib-2 nle-le nless-le not-less-eq* **by** *metis*

lemma *ge-two-fib-unique-idx*: $fib\ i \geq 2 \Longrightarrow fib\ i = fib\ j \Longrightarrow i = j$
using *ge-two-eq-fib-implies-eq-idx* **by** *auto*

lemma *no-fib-lower-bound*: $\neg$ *is-fib n* $\Longrightarrow$ $n \geq 4$
proof(*rule ccontr*)
  assume $\neg$ *is-fib n* $\neg$ $4 \leq n$
  hence $n \in \{0,1,2,3\}$ **by** *auto*
  have *is-fib 0 is-fib 1 is-fib 2 is-fib 3*
    using *fib0 fib1 fib-values fib-implies-is-fib* **by** *blast+*
  then show *False*
    using ‹$\neg$ *is-fib n*› ‹$n \in \{0,1,2,3\}$› **by** *blast*
qed

lemma *pos-fib-has-idx-ge-two*: $n > 0 \Longrightarrow$ *is-fib n* $\Longrightarrow (\exists\ i.\ i \geq 2 \wedge fib\ i = n)$
  unfolding *is-fib-def* **by** (*metis One-nat-def fib-2 fib-mono less-eq-Suc-le nle-le*)

lemma *finite-fib0-idx*: *finite*($\{i.\ fib\ i = 0\}$)
  using *zero-fib-unique-idx finite-nat-set-iff-bounded* **by** *auto*

lemma *finite-fib1-idx*: *finite*($\{i.\ fib\ i = 1\}$)
  using *one-fib-idxs finite-nat-set-iff-bounded* **by** *auto*

lemma *finite-fib-ge-two-idx*: $n \geq 2 \Longrightarrow$ *finite*($\{i.\ fib\ i = n\}$)
  using *ge-two-fib-unique-idx finite-nat-set-iff-bounded* **by** *auto*

lemma *finite-fib-index*: *finite*($\{i.\ fib\ i = n\}$)
  using *finite-fib0-idx finite-fib1-idx finite-fib-ge-two-idx* **by**(*rule nat-induct2*, *auto*)

lemma *no-fib-implies-zero-in-le-idx-set*: $\neg$ *is-fib n* $\Longrightarrow 0 \in \{i.\ fib\ i < n\}$
  using *no-fib-lower-bound* **by** *fastforce*

lemma *no-fib-implies-le-fib-idx-set*: $\neg$ *is-fib n* $\Longrightarrow \{i.\ fib\ i < n\} \neq \{\}$
  using *no-fib-implies-zero-in-le-idx-set* **by** *blast*

lemma *finite-smaller-fibs*: *finite*($\{i.\ fib\ i < n\}$)
proof(*induct n*)
  case (*Suc n*)
  moreover have $\{i.\ fib\ i < Suc\ n\} = \{i.\ fib\ i < n\} \cup \{i.\ fib\ i = n\}$ **by** *auto*
  moreover have *finite*($\{i.\ fib\ i = n\}$) **using** *finite-fib-index* **by** *auto*
  ultimately show *?case* **by** *auto*
qed *simp*

lemma *nat-ge-2-fib-idx-bound*: $2 \leq n \Longrightarrow fib\ i \leq n \Longrightarrow n < fib\ (Suc\ i) \Longrightarrow 2 \leq i$
  by (*metis One-nat-def fib-1 fib-2 le-Suc-eq less-2-cases linorder-not-le not-less-eq*)

lemma *inc-seq-on-aux*: *inc-seq-on c* $\{0..k - 1\} \Longrightarrow n - fib\ i < fib\ (i-1) \Longrightarrow fib$ $(c\ k) < fib\ i \Longrightarrow$
$$(n - fib\ i) = (\sum\ i{=}0..k.\ fib\ (c\ i)) \Longrightarrow Suc\ (c\ k) < i$$
  by (*metis fib-mono bot-nat-0.extremum diff-Suc-1 leD le-SucE linorder-le-less-linear not-add-less1 sum.last-plus*)

lemma *inc-seq-zero-at-start*: *inc-seq-on c* $\{0..k-1\} \Longrightarrow c\ k = 0 \Longrightarrow k = 0$
unfolding *inc-seq-on-def*
by (*metis One-nat-def Suc-pred atLeast0AtMost atMost-iff less-nat-zero-code not-gr-zero order.refl*)

lemma *fib-sum-zero-equiv*: $(\sum\ i{=}n..m{::}nat\ .\ fib\ (c\ i)) = 0 \longleftrightarrow (\forall\ i{\in}\{n..m\}.\ c\ i = 0)$
using *finite-atLeastAtMost sum-eq-0-iff zero-fib-equiv* **by** *auto*

lemma *fib-idx-ge-two-fib-sum-not-zero*: $n \le m \Longrightarrow \forall\, i{\in}\{n..m{::}nat\}.\ c\ i \ge 2 \Longrightarrow \neg\ (\sum\ i{=}n..m.\ fib\ (c\ i)) = 0$
by (*rule ccontr*, *simp add*: *fib-sum-zero-equiv*)

lemma *one-unique-fib-sum*: *inc-seq-on c* $\{0..k-1\} \Longrightarrow \forall\, i{\in}\{0..k\}.\ c\ i \ge 2 \Longrightarrow (\sum\ i{=}0..k.\ fib\ (c\ i)) = 1 \longleftrightarrow k = 0 \land c\ 0 = 2$
proof
assume *assms*: $(\sum i = 0..k.\ fib\ (c\ i)) = 1$ *inc-seq-on c* $\{0..k-1\}$ $\forall\, i{\in}\{0..k\}.\ c\ i \ge 2$
hence $fib\ (c\ 0) + (\sum i = 1..k.\ fib\ (c\ i)) = 1$ **by** (*simp add*: *sum.atLeast-Suc-atMost*)
moreover have $fib\ (c\ 0) \ge 1$ **using** *assms fib-neq-0-nat*[*of c 0*] **by** *force*
ultimately show $k = 0 \land c\ 0 = 2$
using *fib-idx-ge-two-fib-sum-not-zero*[*of 1 k c*] *assms add-is-1 one-fib-idxs* **by**(*cases* $k{=}0$, *fastforce*, *auto*)
qed *simp*

lemma *no-fib-betw-fibs*:
assumes ¬ *is-fib n*
shows $\exists\ i.\ fib\ i < n \land n < fib\ (Suc\ i)$
proof −
have *finite-le-fib*: *finite* $\{i.\ fib\ i < n\}$ **using** *finite-smaller-fibs* **by** *auto*
obtain *i* **where** *max-def*: $i = Max\ \{i.\ fib\ i < n\}$ **by** *blast*
show $\exists\ i.\ fib\ i < n \land n < fib\ (Suc\ i)$
proof(*intro exI conjI*)
have $(Suc\ i) \notin \{i.\ fib\ i < n\}$ **using** *max-def Max-ge Suc-n-not-le-n finite-le-fib* **by** *blast*
thus $fib\ (Suc\ i) > n$
using ‹¬ *is-fib n*› *fib-implies-is-fib linorder-less-linear* **by** *blast*
qed(*insert max-def Max-in* ‹¬ *is-fib n*› *finite-le-fib no-fib-implies-le-fib-idx-set*, *auto*)
qed

lemma *betw-fibs*:
shows $\exists\ i.\ fib\ i \le n \land fib(Suc\ i) > n$
proof(*cases is-fib n*)
case *True*
then obtain *i* **where** *a*: $n = fib\ i$ **unfolding** *is-fib-def* **by** *auto*
then show *?thesis*
by (*metis fib1 Suc-le-eq fib-2 fib-mono fib-strict-mono le0 le-eq-less-or-eq not-less-eq-eq*)
qed(*insert no-fib-betw-fibs*, *force*)

Proof that the sum of non-consecutive Fibonacci numbers with largest member F_i is strictly less then F_{i+1}. This lemma is used for the uniqueness proof.

```
lemma fib-sum-upper-bound:
  assumes inc-seq-on c {0..k−1} ∀ i∈{0..k}. c i ≥ 2
  shows (∑ i=0..k. fib (c i)) < fib (Suc (c k))
proof(insert assms, induct c k arbitrary: k rule: nat-less-induct)
  case 1
  then show ?case
  proof(cases c k)
    case (Suc nat)
    show ?thesis
    proof(cases k)
      case k-Suc: (Suc -)
      hence ck-bounds: c(k−1) + 1 < c k c(k−1) < c k
        using 1(2) unfolding inc-seq-on-def by (force)+
     moreover have (∑ i = 0..k. fib (c i)) = fib(c k) + (∑ i = 0..k−1. fib (c i))

        using k-Suc by simp
      moreover have (∑ i = 0..(k−1). fib (c i)) < fib (Suc (c (k−1)))
        using ck-bounds(2) 1 unfolding inc-seq-on-def by auto
      ultimately show ?thesis
        using Suc smaller-index-implies-fib-le by fastforce
    qed(simp add: fib-index-strict-mono assms(2))
  qed(insert inc-seq-zero-at-start[OF 1(2)], auto)
qed

lemma last-fib-sum-index-constraint:
  assumes n ≥ 2 n = (∑ i=0..k. fib (c i)) inc-seq-on c {0..k−1}
  assumes ∀ i∈{0..k}. c i ≥ 2 fib i ≤ n fib(Suc i) > n
  shows c k = i
proof −
  have 2 ≤ i using assms(1,5,6) nat-ge-2-fib-idx-bound by simp
  have c k > i ⟶ False
    using smaller-index-implies-fib-le assms
    by (metis bot-nat-0.extremum leD sum.last-plus trans-le-add1)
  moreover have c k < i ⟶ False
  proof
    assume c k < i
    have seq: inc-seq-on c {0..k − 1 − 1} ∀ i∈{0..k−1}. 2 ≤ c i
      using assms unfolding inc-seq-on-def by simp+
    have k > 0
      by(rule ccontr, insert ‹c k < i› assms fib-index-strict-mono leD, auto)
    hence c (k−1) + 1 < c k c (k−1) + 3 ≤ i
      using ‹c k < i› assms unfolding inc-seq-on-def by force+
    have (∑ i = 0..k−1. fib (c i)) + fib (c k) = (∑ i = 0..k. fib (c i))
      using sum.atLeast0-atMost-Suc Suc-pred'[OF ‹k > 0›] by metis
    moreover have fib (Suc (c (k−1))) ≤ fib (i−2)
      using ‹c k < i› ‹c (k−1) + 1 < c k› by (simp add: fib-mono)
    moreover have fib (c k) ≤ fib (i−1)
```

using ‹$c\ k < i$› *fib-mono* **by** *fastforce*
ultimately have $(\sum i = 0..k.\ fib\ (c\ i)) < fib\ (i-1) + fib\ (i-2)$
using *assms* ‹$c\ k < i$› ‹$k > 0$› *fib-sum-upper-bound*[*OF seq*(*1*) *seq*(*2*)] **by** *simp*
hence $(\sum i = 0..k.\ fib\ (c\ i)) < fib\ i$
using *fib.simps*(*3*)[*of* $i-2$] *assms*(*4*) ‹$c\ k < i$›
by (*metis add-2-eq-Suc diff-Suc-1* ‹$2 \le i$› *le-add-diff-inverse*)
then show *False*
using *assms* **by** *simp*
qed
ultimately show *?thesis* **by** *simp*
qed

1.3 Theorem

Now, both parts of Zeckendorf's Theorem can be proven. Firstly, the existence of an increasing sequence for a positive integer N such that the corresponding Fibonacci numbers sum up to N is proven. Then, the uniqueness of such an increasing sequence is proven.

lemma *fib-implies-zeckendorf*:
assumes *is-fib n* $n > 0$
shows $\exists\ c\ k.\ n = (\sum\ i{=}0..k.\ fib(c\ i)) \land$ *inc-seq-on c* $\{0..k-1\} \land (\forall\ i{\in}\{0..k\}.\ c\ i \ge 2)$
proof −
from *assms* **obtain** *i* **where** *i-def*: $fib\ i = n$ $i \ge 2$ **using** *pos-fib-has-idx-ge-two* **by** *auto*
define *c* **where** *c-def*: $(c :: nat \Rightarrow nat) = (\lambda\ n{::}nat.\ if\ n = 0\ then\ i\ else\ i + 3)$
from *i-def* **have** $n = (\sum i = 0..0.\ fib\ (c\ i))$ **by** (*simp add*: *c-def*)
moreover have *inc-seq-on c* $\{0..0\}$ **by** (*simp add*: *c-def inc-seq-on-def*)
ultimately show $\exists\ c\ k.\ n = (\sum\ i{=}0..k.\ fib(c\ i)) \land$ *inc-seq-on c* $\{0..k-1\} \land (\forall\ i{\in}\{0..k\}.\ c\ i \ge 2)$
using *i-def c-def* **by** *fastforce*
qed

theorem *zeckendorf-existence*:
assumes $n > 0$
shows $\exists\ c\ k.\ n = (\sum\ i{=}0..k.\ fib\ (c\ i)) \land$ *inc-seq-on c* $\{0..k-1\} \land (\forall\ i{\in}\{0..k\}.\ c\ i \ge 2)$
using *assms*
proof(*induct n rule*: *nat-less-induct*)
case (*1 n*)
then show *?case*
proof(*cases is-fib n*)
case *False*
obtain *i* **where** *bounds*: $fib\ i < n$ $n < fib\ (Suc\ i)$ $i > 0$
using *no-fib-betw-fibs 1*(*2*) *False* **by** *force*
then obtain *c k* **where** *seq*: $(n - fib\ i) = (\sum\ i{=}0..k.\ fib\ (c\ i))$ *inc-seq-on c* $\{0..k-1\}$ $\forall\ i{\in}\{0..k\}.\ c\ i \ge 2$

```
      using 1 fib-neq-0-nat zero-less-diff diff-less by metis
    let ?c' = (λ n. if n = k+1 then i else c n)
    have diff-le-fib: n − fib i < fib(i−1)
      using bounds fib2 not0-implies-Suc[of i] by auto
    hence ck-lt-fib: fib (c k) < fib i
      using fib-Suc-mono[of i−1] bounds by (simp add: sum.last-plus seq)
    have inc-seq-on ?c' {0..k}
      using inc-seq-on-aux[OF seq(2) diff-le-fib ck-lt-fib seq(1)] One-nat-def
            inc-seq-on-def leI seq by force
    moreover have n = (∑ i=0..k+1. fib (?c' i))
      using bounds seq by simp
    moreover have ∀ i ∈ {0..k+1}. ?c' i ≥ 2
      using seq bounds fib2 not0-implies-Suc[of i] atLeastAtMost-iff
            diff-is-0-eq' less-nat-zero-code not-less-eq-eq by fastforce
    ultimately show ?thesis by fastforce
  qed(insert fib-implies-zeckendorf, auto)
qed

lemma fib-unique-fib-sum:
  fixes k :: nat
  assumes n ≥ 2 inc-seq-on c {0..k−1} ∀ i∈{0..k}. c i ≥ 2
  assumes n = fib i
  shows n = (∑ i=0..k. fib (c i)) ⟷ k = 0 ∧ c 0 = i
proof
  assume ass: n = (∑ i = 0..k. fib (c i))
  obtain j where bounds: fib j ≤ n fib(Suc j) > n j ≥ 2
    using betw-fibs assms nat-ge-2-fib-idx-bound by blast
  have idx-eq: c k = j
    using last-fib-sum-index-constraint assms(1−3) ass bounds by simp
  have i = j
    using bounds ass assms
   by (metis Suc-leI fib-mono ge-two-fib-unique-idx le-neq-implies-less linorder-not-le)
  have k > 0 ⟶ fib i = fib i + (∑ i = 0..k−1. fib (c i))
     using ass assms by (metis idx-eq One-nat-def Suc-pred ‹i = j› add.commute
sum.atLeast0-atMost-Suc)
  hence k > 0 ⟶ False
    using fib-idx-ge-two-fib-sum-not-zero[of 0 k−1 c] assms by auto
  then show k = 0 ∧ c 0 = i using ‹i = j› idx-eq by simp
qed(auto simp: assms)

theorem zeckendorf-unique:
  assumes n > 0
  assumes n = (∑ i=0..k. fib (c i)) inc-seq-on c {0..k−1} ∀ i∈{0..k}. c i ≥ 2
  assumes n = (∑ i=0..k'. fib (c' i)) inc-seq-on c' {0..k'−1} ∀ i∈{0..k'}. c' i ≥
2
  shows k = k' ∧ (∀ i ∈ {0..k}. c i = c' i)
  using assms
proof(induct n arbitrary: k k' rule: nat-less-induct)
  case IH: (1 n)
```

```
  consider n = 0 | n = 1 | n ≥ 2 by linarith
  then show ?case
  proof(cases)
    case 3
    obtain i where bounds: fib i ≤ n fib(Suc i) > n 2 ≤ i
      using betw-fibs nat-ge-2-fib-idx-bound 3 by blast
    have last-idx-eq: c′ k′ = i c k = i c′ k′ = c k
        using last-fib-sum-index-constraint[OF 3] IH(6−8) IH(3−5) bounds by
blast+
    then show ?thesis
    proof(cases is-fib n)
      case True
      hence fib i = n
      unfolding is-fib-def using bounds IH(2−8) fib-mono leD nle-le not-less-eq-eq
by metis
      hence k = 0 c 0 = i k′ = 0 c′ 0 = i
        using fib-unique-fib-sum 3 IH(3−8) by metis+
        then show ?thesis by simp
    next
      case False
      have k > 0
        using IH(3) False unfolding is-fib-def by fastforce
      have k′ > 0
        using IH(6) False  unfolding is-fib-def by fastforce
     have 0 < n − fib (c k) using False bounds last-idx-eq(2) unfolding is-fib-def
by fastforce
      moreover have n − fib (c k) < n
       using bounds last-idx-eq by (simp add: dual-order.strict-trans1 fib-neq-0-nat)
      moreover have n − fib (c k) = (∑ i = 0..k−1. fib (c i))
         using sum.atLeast0-atMost-Suc[of λ i. fib (c i) k−1] Suc-diff-1 ‹k > 0›
IH(3) by simp
      moreover have n − fib (c′ k′ ) = (∑ i = 0..k′−1. fib (c′ i))
        using sum.atLeast0-atMost-Suc[of λ i. fib (c′ i) k′−1] Suc-diff-1 ‹k′ > 0›
IH(6) by simp
      moreover have inc-seq-on c {0..k−1 − 1} ∀ i∈{0..k−1}. 2 ≤ c i
        using IH(4,5) unfolding inc-seq-on-def by auto
      moreover have inc-seq-on c′ {0..k′−1 − 1} ∀ i∈{0..k′−1}. 2 ≤ c′ i
        using IH(7,8) unfolding inc-seq-on-def by auto
      ultimately have k−1 = k′−1 ∧ (∀ i∈{0..k−1}. c i = c′ i)
        using IH(1) unfolding last-idx-eq by blast
      then show ?thesis
        using IH(1) last-idx-eq by (metis One-nat-def Suc-pred ‹0 < k′› ‹0 < k›
atLeastAtMost-iff le-Suc-eq)
    qed
  qed(insert IH one-unique-fib-sum, auto)
qed

end
```

References

[1] C. G. Lekkerkerker. Voorstelling van natuurlijke getallen door een som van getallen van fibonacci. *Stichting Mathematisch Centrum. Zuivere Wiskunde*, (ZW 30/51), 1951.