Euler’s Partition Theorem

Lukas Bulwahn

September 13, 2023

Abstract

Euler’s Partition Theorem states that the number of partitions with only distinct parts is equal to the number of partitions with only odd parts. The combinatorial proof follows John Harrison’s pre-existing HOL Light formalization [1]. To understand the rough idea of the proof, I read the lecture notes of the MIT course 18.312 on Algebraic Combinatorics [2] by Gregg Musiker. This theorem is the 45th theorem of the Top 100 Theorems list.

Contents

1 Euler’s Partition Theorem 1
  1.1 Preliminaries .............................................. 1
    1.1.1 Additions to Divides Theory ......................... 1
    1.1.2 Additions to Groups-Big Theory .................... 2
    1.1.3 Additions to Finite-Set Theory ..................... 2
    1.2 Binary Encoding of Natural Numbers ................. 2
    1.3 Decomposition of a Number into a Power of Two and an Odd Number .................................. 3
    1.4 Partitions With Only Distinct and Only Odd Parts ..... 4
    1.5 Euler’s Partition Theorem .............................. 5

1 Euler’s Partition Theorem

theory Euler-Partition
imports
  Main
  Card-Number-Partitions.Number-Partition
begin

1.1 Preliminaries

1.1.1 Additions to Divides Theory

lemma power-div-nat:
assumes $c \leq b$
assumes $a > 0$
shows $(a :: \text{nat}) \sim b \div a \sim c = a \sim (b - c)$
⟨proof⟩

1.1.2 Additions to Groups-Big Theory

lemma sum-div:
assumes finite $A$
assumes $∀ a. a ∈ A \implies (b :: 'b::euclidean-semiring) \mid a$
shows $(∑ a ∈ A. f a) \div b = (∑ a ∈ A. (f a) \div b)$
⟨proof⟩

lemma sum-mod:
assumes finite $A$
assumes $∀ a. a ∈ A \implies f a \mod b = (0 :: 'b::unique-euclidean-semiring)$
shows $(∑ a ∈ A. f a) \mod b = 0$
⟨proof⟩

1.1.3 Additions to Finite-Set Theory

lemma finite-exponents:
finite \{ i. 2 ^ i \leq (n :: \text{nat}) \}
⟨proof⟩

1.2 Binary Encoding of Natural Numbers

definition bitset :: \text{nat} ⇒ \text{nat set}
where
bitset $n = \{ i. \text{odd} (n \div (2 ^ i)) \}$

lemma in-bitset-bound:
b ∈ bitset $n \implies 2 ^ b \leq n$
⟨proof⟩

lemma in-bitset-bound-weak:
b ∈ bitset $n \implies b \leq n$
⟨proof⟩

lemma finite-bitset:
finite (bitset $n$)
⟨proof⟩

lemma bitset-0:
bitset $0 = \{ \}$
⟨proof⟩

lemma bitset-2n: bitset $(2 * n) = \text{Suc} ^ ' (bitset n)$
⟨proof⟩
lemma \texttt{bitset-Suc}:
\textbf{assumes} even \texttt{n} \\
\textbf{shows} \texttt{bitset} (\texttt{n} + 1) = \texttt{insert 0} (\texttt{bitset} \texttt{n})  \\
\langle \text{proof} \rangle

lemma \texttt{bitset-2n1}:
\texttt{bitset} (2 \times \texttt{n} + 1) = \texttt{insert 0} (\texttt{Suc} \cdot (\texttt{bitset} \texttt{n}))  \\
\langle \text{proof} \rangle

lemma \texttt{sum-bitset}:
\langle \sum \texttt{i} \in \texttt{bitset} \texttt{n}. 2 ^ \texttt{i} \rangle = \texttt{n}  \\
\langle \text{proof} \rangle

lemma \texttt{binarysum-div}:
\textbf{assumes} finite \texttt{B} \\
\textbf{shows} \langle \sum \texttt{i} \in \texttt{B}. (2 ^ \cdot \texttt{nat}) ^ \texttt{i} \rangle \texttt{div} 2 ^ \texttt{j} = \langle \sum \texttt{i} \in \texttt{B}. \texttt{if} \texttt{i} < \texttt{j} \texttt{then} 0 \texttt{else} 2 ^ (\texttt{i} - \texttt{j}) \rangle  \\
\langle \text{proof} \rangle

lemma \texttt{odd-iff}:
\textbf{assumes} finite \texttt{B} \\
\textbf{shows} \texttt{odd} \langle \sum \texttt{i} \in \texttt{B}. \texttt{if} \texttt{i} < \texttt{x} \texttt{then} 0 ^ \cdot \texttt{nat} \texttt{else} 2 ^ (\texttt{i} - \texttt{x}) \rangle = (\texttt{x} \in \texttt{B}) \texttt{(is odd} \langle \sum \texttt{i} \in -. \texttt{?s i} \rangle = -. \rangle)  \\
\langle \text{proof} \rangle

lemma \texttt{bitset-sum}:
\textbf{assumes} finite \texttt{B} \\
\textbf{shows} \texttt{bitset} \langle \sum \texttt{i} \in \texttt{B}. 2 ^ \texttt{i} \rangle = \texttt{B}  \\
\langle \text{proof} \rangle

1.3 \quad \textbf{Decomposition of a Number into a Power of Two and an Odd Number}

\textbf{function} (\texttt{sequential}) \texttt{index :: nat} \Rightarrow \texttt{nat} \\
\textbf{where}  \\
\texttt{index 0} = 0  \\
| \texttt{index} \texttt{n} = (\texttt{if odd} \texttt{n} \texttt{then} 0 \texttt{else} \texttt{Suc} (\texttt{index} \,(\texttt{n} \texttt{div} 2)))  \\
\langle \text{proof} \rangle

\textbf{termination}  \\
\langle \text{proof} \rangle

\textbf{function} (\texttt{sequential}) \texttt{oddpart :: nat} \Rightarrow \texttt{nat} \\
\textbf{where}  \\
\texttt{oddpart 0} = 0  \\
| \texttt{oddpart} \texttt{n} = (\texttt{if odd} \texttt{n} \texttt{then} \texttt{n} \texttt{else} \texttt{oddpart} \,(\texttt{n} \texttt{div} 2))  \\
\langle \text{proof} \rangle
termination
⟨proof⟩

lemma odd-oddpart:
  odd (oddpart n) ⟷ n ≠ 0
⟨proof⟩

lemma index-oddpart-decomposition:
  n = 2 ^ (index n) * oddpart n
⟨proof⟩

lemma oddpart-leq:
  oddpart n ≤ n
⟨proof⟩

lemma index-oddpart-unique:
  assumes odd (m :: nat) odd m'
  shows (2 ^ i * m = 2 ^ i' * m') ⟷ (i = i' ∧ m = m')
⟨proof⟩

lemma index-oddpart:
  assumes odd m
  shows index (2 ^ i * m) = i oddpart (2 ^ i * m) = m
⟨proof⟩

1.4 Partitions With Only Distinct and Only Odd Parts

definition odd-of-distinct :: (nat ⇒ nat) ⇒ nat ⇒ nat
where
  odd-of-distinct p = (λi. if odd i then (∑ j | p (2 ^ j * i) = 1. 2 ^ j) else 0)

definition distinct-of-odd :: (nat ⇒ nat) ⇒ nat ⇒ nat
where
  distinct-of-odd p = (λi. if index i ∈ bitset (p (oddpart i)) then 1 else 0)

lemma odd:
  odd-of-distinct p i ≠ 0 ⟹ odd i
⟨proof⟩

lemma distinct-distinct-of-odd:
  distinct-of-odd p i ≤ 1
⟨proof⟩

lemma odd-of-distinct:
  assumes odd-of-distinct p i ≠ 0
  assumes ∀i. p i ≠ 0 ⟹ i ≤ n
  shows 1 ≤ i ∧ i ≤ n
⟨proof⟩
lemma distinct-of-odd:
  assumes $\forall i. \ p \ i \leq n \land i. \ p \ i \neq 0 \implies odd \ i$
  assumes distinct-of-odd \ $p \ i \neq 0$
  shows $1 \leq i \land i \leq n$
⟨proof⟩

lemma odd-distinct:
  assumes $\forall i. \ p \ i \neq 0 \implies odd \ i$
  shows odd-of-distinct \ (distinct-of-odd \ $p$) = $p$
⟨proof⟩

lemma distinct-odd:
  assumes $\forall i. \ p \ i \neq 0 \implies 1 \leq i \land i \leq n \land i. \ p \ i \leq 1$
  shows distinct-of-odd \ (odd-of-distinct \ $p$) = $p$
⟨proof⟩

lemma sum-distinct-of-odd:
  assumes $\forall i. \ p \ i \neq 0 \implies 1 \leq i \land i \leq n$
  assumes $\forall i. \ p \ i \leq n$
  assumes $\forall i. \ p \ i \neq 0 \implies odd \ i$
  shows $(\sum i \leq n. \ distinct-of-odd \ p \ i \ * \ i) = (\sum i \leq n. \ p \ i \ * \ i)$
⟨proof⟩

lemma leq-n:
  assumes $\forall i. \ 0 < p \ i \implies 1 \leq i \land i \leq (n::nat)$
  assumes $(\sum i \leq n. \ p \ i \ * \ i) = n$
  shows $p \ i \ * \ i \leq n$
⟨proof⟩

lemma distinct-of-odd-in-distinct-partitions:
  assumes $p \in \{p. \ p \ partitions \ n \land (\forall i. \ p \ i \neq 0 \implies odd \ i)\}$
  shows distinct-of-odd \ $p \in \{p. \ p \ partitions \ n \land (\forall i. \ p \ i \leq 1)\}$
⟨proof⟩

lemma odd-of-distinct-in-odd-partitions:
  assumes $p \in \{p. \ p \ partitions \ n \land (\forall i. \ p \ i \leq 1)\}$
  shows odd-of-distinct \ $p \in \{p. \ p \ partitions \ n \land (\forall i. \ p \ i \neq 0 \implies odd \ i)\}$
⟨proof⟩

1.5 Euler’s Partition Theorem

theorem Euler-partition-theorem:
  card \ $\{p. \ p \ partitions \ n \land (\forall i. \ p \ i \leq 1)\} = card \ \{p. \ p \ partitions \ n \land (\forall i. \ p \ i \neq 0 \implies odd \ i)\}$
  (is card ?distinct-partitions = card ?odd-partitions)
⟨proof⟩

end
References
