Cardinality of Set Partitions

Lukas Bulwahn

August 16, 2018

Abstract

The theory’s main theorem states that the cardinality of set partitions of size \( k \) on a carrier set of size \( n \) is expressed by Stirling numbers of the second kind. In Isabelle, Stirling numbers of the second kind are defined in the AFP entry ‘Discrete Summation’ [1] through their well-known recurrence relation. The main theorem relates them to the alternative definition as cardinality of set partitions. The proof follows the simple and short explanation in Richard P. Stanley’s ‘Enumerative Combinatorics: Volume 1’ [2] and Wikipedia [3], and unravels the full details and implicit reasoning steps of these explanations.

Contents

1 Set Partitions
   1.1 Useful Additions to Main Theories ......................... 2
   1.2 Introduction and Elimination Rules ........................ 2
   1.3 Basic Facts on Set Partitions ............................. 2
   1.4 The Unique Part Containing an Element in a Set Partition . 4
   1.5 Cardinality of Parts in a Set Partition ................... 7
   1.6 Operations on Set Partitions ............................. 8

2 Combinatorial Basics
   2.1 Preliminaries ....................................... 12
      2.1.1 Injectivity and Disjoint Families .................. 12
      2.1.2 Cardinality Theorems for Set.bind ................. 12
   2.2 Third Version of Injectivity Solver ....................... 13

3 Cardinality of Set Partitions .............................. 15

1 Set Partitions

theory Set-Partition
imports
   HOL-Library.Disjoint-Sets
1.1 Useful Additions to Main Theories

lemma set-eqI':
  assumes \( \forall x. x \in A \Rightarrow x \in B \) 
  assumes \( \forall x. x \in B \Rightarrow x \in A \) 
  shows \( A = B \) 
using assms by auto

lemma comp-image:
  \(((' f o (' g)) = (' (f o g))\)
by rule auto

1.2 Introduction and Elimination Rules

The definition of \(\text{partition-on}\) is in HOL−Library.Disjoint-Sets.

lemma partition-onI:
  assumes \( \forall p. p \in P \Rightarrow p \neq \{} \) 
  assumes \( \bigcup P = A \) 
  assumes \( \forall p p'. p \in P \Rightarrow p' \in P \Rightarrow p \neq p' \Rightarrow p \cap p' = \{} \) 
  shows \(\text{partition-on} A P\) 
using assms unfolding partition-on-def disjoint-def by blast

lemma partition-onE:
  assumes \(\text{partition-on} A P\) 
  obtains \( \forall p. p \in P \Rightarrow p \neq \{} \) 
  \( \bigcup P = A \) 
  \( \forall p p'. p \in P \Rightarrow p' \in P \Rightarrow p \neq p' \Rightarrow p \cap p' = \{} \) 
using assms unfolding partition-on-def disjoint-def by blast

1.3 Basic Facts on Set Partitions

lemma partition-onD4: \(\text{partition-on} A P \Rightarrow p \in P \Rightarrow q \in P \Rightarrow x \in p \Rightarrow x \in q \Rightarrow p = q\) 
by (auto simp: partition-on-def disjoint-def)

lemma partition-subset-imp-notin:
  assumes \(\text{partition-on} A P X \in P\) 
  assumes \( X' \subset X \) 
  shows \( X' \notin P \) 
proof
  assume \( X' \in P \) 
  from \( X' \in P \), \(\text{partition-on} A P\) have \( X' \neq \}\) 
  using partition-onD3 by blast 
  moreover from \( X' \in P \), \( X \in P \), \(\text{partition-on} A P\), \( X' \subset X \) have disjoint X X' 
  by (metis disjoint-def disjointD inf.strict-order-iff partition-onD2)

2
moreover note \( X' \subseteq X \)
ultimately show False
  by (meson all-not-in-conv disjoint-iff psubsetD)
qed

lemma partition-on-Diff:
  assumes P: partition-on A P shows Q \subseteq P \implies partition-on (A - \bigcup Q) (P - Q)
  using P P[THEN partition-onD4] by (auto simp: partition-on-def disjoint-def)

lemma partition-on-UN:
  assumes A: partition-on A B and B: \( \forall b. b \in B \implies \text{partition-on } b \) (P b)
  shows partition-on A (\bigcup b\in B. P b)
  proof (rule partition-onI)
    show \( \bigcup (\bigcup b\in B. P b) = A \)
      using B[THEN partition-onD1] A[THEN partition-onD1] by blast
  next
    show p \neq {} if p \in (\bigcup b\in B. P b) for p
      using B[THEN partition-onD3] that by auto
  next
    fix p q assume p \in (\bigcup i\in B. P i) q \in (\bigcup i\in B. P i) and p \neq q
    then obtain i j where i: p \in P i i \in B and j: q \in P j j \in B
      by auto
    show p \cap q = {}
      proof cases
        assume i = j then show ?thesis
          using i j \( p \neq q \) B[THEN partition-onD2, of i] by (simp add: disjointD)
      next
        assume i \neq j
        then have disjoint i j
          using i j A[THEN partition-onD2] by (auto simp: pairwise-def)
        moreover have p \subseteq i q \subseteq j
          using B[THEN partition-onD1, of i, symmetric] B[THEN partition-onD1, of j, symmetric] i j by auto
        ultimately show ?thesis
          by (auto simp: disjoint-def)
      qed
    qed

lemma partition-on-notemptyI:
  assumes partition-on A P
  assumes A \neq {}
  shows P \neq {}
  using assms by (auto elim: partition-onE)

lemma partition-on-disjoint:
  assumes partition-on A P
  assumes partition-on B Q
  assumes A \cap B = {}


shows $P \cap Q = \{\}$
using assms by (fastforce elim: partition-onE)

lemma partition-on-eq-implies-eq-carrier:
  assumes partition-on $A \mid Q$
  assumes partition-on $B \mid Q$
  shows $A = B$
using assms by (fastforce elim: partition-onE)

lemma partition-on-insert:
  assumes partition-on $A \mid P$
  assumes disjoint $A \mid X \mid X \neq \{\}$
  assumes $A \cup X = A'$
  shows partition-on $A' \mid (insert X P)$
using assms by (auto simp: partition-on-def disjoint-def disjnt-def)


lemma partition-on-insert-
  assumes partition-on $(A - X) \mid P$
  assumes $X \subseteq A \mid X \neq \{\}$
  shows partition-on $A \mid (insert X P)$
proof --
  have disjoint $(A - X) \mid X$ by (simp add: disjoint)
  from assms(1) this assms(3) have partition-on $((A - X) \cup X) \mid (insert X P)$
    by (auto intro: partition-on-insert)
  from this $\langle X \subseteq A \rangle$ show ?thesis
    by (metis Diff-partition sup-commute)
qed

lemma partition-on-insert-singleton:
  assumes partition-on $A \mid P \mid a \notin A insert a \mid A = A'$
  shows partition-on $A' \mid (insert \{a\} \mid P)$
using assms by (auto simp: partition-on-def disjoint-def disjnt-def)

lemma partition-on-remove-singleton:
  assumes partition-on $A \mid P X \in P A \mid X = A'$
  shows partition-on $A' \mid (P - \{X\})$
using assms partition-on-Diff by (metis Diff-cancel Diff-subset cSup-singleton insert-subset)

1.4 The Unique Part Containing an Element in a Set Partition

lemma partition-on-partition-on-unique:
  assumes partition-on $A \mid P$
  assumes $x \in A$
  shows $\exists X. x \in X \land X \in P$
proof --
  from $\langle partition-on A \mid P \rangle$ have $\bigcup P = A$
qed
by (auto elim; partition-onE)
from this \( x \in A \) obtain \( X \) where \( X: x \in X \land X \in P \) by blast
{
  fix \( Y \)
  assume \( x \in Y \land Y \in P \)
  from this have \( X = Y \)
  using \( X \) (partition-on A P) by (meson partition-onE disjoint-iff-not-equal)
}
from this \( X \) show ?thesis by auto
qed

lemma partition-on-the-part-mem:
  assumes partition-on A P
  assumes \( x \in A \)
  shows \( (\text{T}HE \ X. \ x \in X \land X \in P) \in P \)
proof –
  from \( x \in A \) have \( \exists! X. \ x \in X \land X \in P \)
  using (partition-on A P) by (simp add: partition-on-partition-on-unique)
from this show \( (\text{T}HE \ X. \ x \in X \land X \in P) \in P \)
  by (metis (no-types, lifting) theI)
qed

lemma partition-on-in-the-unique-part:
  assumes partition-on A P
  assumes \( x \in A \)
  shows \( x \in (\text{T}HE \ X. \ x \in X \land X \in P) \)
proof –
  from assms have \( \exists! X. \ x \in X \land X \in P \)
  by (simp add: partition-on-partition-on-unique)
from this show ?thesis
  by (metis (mono-tags, lifting) theI′)
qed

lemma partition-on-the-part-eq:
  assumes partition-on A P
  assumes \( x \in X \land X \in P \)
  shows \( (\text{T}HE \ X. \ x \in X \land X \in P) = X \)
proof –
  from \( \langle x \in X \rangle \langle X \in P \rangle \) have \( x \in A \)
  using (partition-on A P) by (auto elim: partition-onE)
from this have \( \exists! X. \ x \in X \land X \in P \)
  using (partition-on A P) by (simp add: partition-on-partition-on-unique)
from \( \langle x \in X \rangle \langle X \in P \rangle \) this show \( (\text{T}HE \ X. \ x \in X \land X \in P) = X \)
  by (auto intro!: the1-equality)
qed

lemma the-unique-part-alternative-def:
  assumes partition-on A P
assumes $x \in A$
shows $(\text{THE } x \in X \land x \in P) = \{y. \exists X \in P. x \in X \land y \in X\}$
proof
  \begin{itemize}
  \item show $(\text{THE } x \in X \land x \in P) \subseteq \{y. \exists X \in P. x \in X \land y \in X\}$
  \end{itemize}
proof
  \begin{itemize}
  \item fix $y$
  \item assume $y \in (\text{THE } x \in X \land x \in P)$
  \item moreover from ($x \in A \land x \in (\text{THE } x \in X \land x \in P)$
    \begin{itemize}
    \item using partition-on $A P$, partition-on-in-the-unique-part by force
    \end{itemize}
  \item moreover from ($x \in A \land x \in (\text{THE } x \in X \land x \in P) \subseteq P$
    \begin{itemize}
    \item using partition-on $A P$, partition-on-the-part-mem by force
    \end{itemize}
  \item ultimately show $y \in \{y. \exists X \in P. x \in X \land y \in X\}$ by auto
  \end{itemize}
qed
next
show \{y. \exists X \in P. x \in X \land y \in X\} \subseteq (\text{THE } x \in X \land x \in P)\)
proof
  \begin{itemize}
  \item fix $y$
  \item assume $y \in \{y. \exists X \in P. x \in X \land y \in X\}$
  \item from this obtain $X$ where $x \in X$ and $y \in X$ and $x \in P$ by auto
  \item from ($x \in X \land x \in P$) have $(\text{THE } x \in X \land x \in P) = X$
    \begin{itemize}
    \item using partition-on $A P$, partition-on-the-part-eq by force
    \end{itemize}
  \item from this ($y \in X$) show $y \in (\text{THE } x \in X \land x \in P)$ by simp
  \end{itemize}
qed

lemma partition-on-all-in-part-eq-part:
  assumes partition-on $A P$
  assumes $X' \in P$
  shows \{x \in A. (\text{THE } x \in X \land x \in P) = X'\} = X'$
proof
  \begin{itemize}
  \item show \{x \in A. (\text{THE } x \in X \land x \in P) = X'\} \subseteq X'$
    \begin{itemize}
    \item using assms(1) partition-on-in-the-unique-part by force
    \end{itemize}
  \item next
  \begin{itemize}
  \item show $X' \subseteq \{x \in A. (\text{THE } x \in X \land x \in P) = X'\}$
    \begin{itemize}
    \item proof
    \begin{itemize}
    \item fix $x$
    \item assume $x \in X'$
    \item from ($x \in X \land x \in P$, have $x \in A$
      \begin{itemize}
      \item using partition-on $A P$ by (auto elim: partition-onE)
      \end{itemize}
    \item moreover from ($x \in X \land x \in P$) have $(\text{THE } x \in X \land x \in P) = X'$
      \begin{itemize}
      \item using partition-on $A P$, partition-on-the-part-eq by fastforce
      \end{itemize}
    \item ultimately show $x \in \{x \in A. (\text{THE } x \in X \land x \in P) = X'\}$ by auto
    \end{itemize}
    \end{itemize}
  \end{itemize}
\end{itemize}
qed

lemma partition-on-part-characteristic:
  assumes partition-on $A P$
  assumes $X \in P \land x \in X$
  shows $X = \{y. \exists X \in P. x \in X \land y \in X\}$
proof –
from \( \{ x \in X \} \land X \in P \) have \( x \in A \)
  using \langle partition-on A P, partition-onE \rangle by blast
from \( \{ x \in X \} \land X \in P \) have \( X = (\text{THE} \ X, x \in X \land X \in P) \)
  using \langle partition-on A P, \rangle by \langle simp add: partition-on-the-part-eq \rangle
also from \( x \in A \) have \( \{ y. \exists X \in P, x \in X \land y \in X \} \)
  using \langle partition-on A P, \rangle by \langle simp add: partition-on-the-part-eq \rangle
using \langle partition-on A P, \rangle by simp

finally show \?thesis .

qed

lemma partition-on-no-partition-outside-carrier:
  assumes partition-on A P
  assumes \( x /\in A \)
  shows \( \{ y. \exists X \in P, x \in X \land y \in X \} = \{} \)
  using assms unfolding partition-on-def by auto

1.5 Cardinality of Parts in a Set Partition

lemma partition-on-le-set-elements:
  assumes finite A
  assumes partition-on A P
  shows \( \text{card} \ P \leq \text{card} \ A \)
  using assms
proof (induct A arbitrary: P)
  case empty
  from this show \( \text{card} \ P \leq \text{card} \{} \) by \langle simp add: partition-on-empty \rangle
next
  case (insert a A)
  show ?case
proof (cases \( \{ a \} \in P \))
    case True
    have prop-partition-on: \( \forall p \in P. \, p \neq \{} \cup P = \text{insert} \ a \ A \)
      \( \forall p \in P. \, \forall p' \in P. \, p \neq p' \longrightarrow p \cap p' = \{} \)
      using \langle partition-on (insert a A) P, \rangle by \langle fastforce elim: partition-onE \rangle+
from this(2, 3) \( a /\in A \) \( \{ a \} \in P \) have A-eq: \( A = \bigcup \{ P - \{ \{ a \} \} \} \)
    by auto \langlemetis Int-iff UnionI empty-iff insert-iff\rangle
from prop-partition-on A-eq have partition-on: \( \text{partition-on} \ A \ (P - \{ \{ a \} \}) \)
    by \langle intro partition-onI \rangle auto
from insert.hyps(3) this have \( \text{card} \ (P - \{ \{ a \} \}) \leq \text{card} \ A \) by simp
from this insert(1, 2, 4) \( \{ a \} \in P \) show \?thesis
  using finite-elements[OF \langle finite A, partition-on \rangle \] by simp
next
  case False
from \langle partition-on (insert a A) P, \rangle obtain \( p \) where p-def: \( p \in P \ a \in p \)
  by \langle blast elim: partition-onE \rangle
from \langle partition-on (insert a A) P, \rangle p-def have a-notmem: \( \forall p' \in P - \{ p \}. \, a /\in p' \)
  by \langle blast elim: partition-onE \rangle
from ⟨partition-on (insert a A) P⟩ p-def have p − {a} /∈ P
unfolding partition-on-def disjoint-def
by (metis Diff-insert-absorb Diff-subset inf.orderE mk-disjoint-insert)
let ?P′ = insert (p − {a}) (P − {p})
have partition-on A ?P′
proof (rule partition-onI)
from ⟨partition-on (insert a A) P⟩ have ∀p∈P. p ≠ {} by (auto elim: partition-onE)
from this p-def ⟨{a} /∈ P⟩ show ⋀p′. p′∈insert (p − {a}) (P − {p}) =⇒ p′ ≠ {}
  by (simp; metis (no-types) Diff-eq-empty-iff subset-singletonD)
next
from ⟨partition-on (insert a A) P⟩ have ⋃P = insert a A by (auto elim: partition-onE)
next
from p-def this ⟨a /∈ A⟩ a-notmem show ⋃(insert (p − {a}) (P − {p}) = A by auto
next
  show ⋀pa pa pa′. pa∈insert (p − {a}) (P − {p}) =⇒ pa′∈insert (p − {a}) (P − {p}) =⇒ pa ≠ pa′ =⇒ pa ∩ pa′ = {}
  using ⟨partition-on (insert a A) P⟩ p-def a-notmem
  unfolding partition-on-def disjoint-def
  by (metis disjoint-iff-not-equal insert-Diff insert-iff)
qed
have finite P using ⟨finite A⟩ ⟨partition-on A ?P⟩ finite-elements by fastforce
have card P = Suc (card (P − {p}))
  using ⟨partition-on A P⟩ card.remove by fastforce
also have ... = card ?P′ using ⟨p − {a} /∈ P⟩ ⟨finite P⟩ by simp
also have ... ≤ card A using ⟨partition-on A ?P⟩ insert.hyps(3) by simp
also have ... ≤ card ⟨insert a A⟩ by (simp add: card-insert-le ⟨finite A⟩ )
finally show ?thesis .
qed
qed

1.6 Operations on Set Partitions

lemma partition-on-union:
assumes A ∩ B = {}
assumes partition-on A P
assumes partition-on B Q
shows partition-on (A ∪ B) (P ∪ Q)
proof (rule partition-onI)
fix X
assume X ∈ P ∪ Q
from this ⟨partition-on A P⟩ ⟨partition-on B Q⟩ show X ≠ {} by (auto elim: partition-onE)
next
show ⋃(P ∪ Q) = A ∪ B
  using ⟨partition-on A P⟩ ⟨partition-on B Q⟩ by (auto elim: partition-onE)
next

8
fix $X$ $Y$
assume $X \in P \cup Q \ Y \in P \cup Q \ X \neq Y$
from this assms show $X \cap Y = \{\}$
by (elim UnE partition-onE) auto
qed

lemma partition-on-split1:
assumes partition-on $A \ (P \cup Q)$
shows partition-on $((\bigcup P) \ P)$
proof (rule partition-onI)
fix $p$
assume $p \in P$
from this assms show $p \neq \{\}$
using Un-iff partition-onE by auto
next
show $\bigcup P = \bigcup P ..$
next
fix $p \ p'$
assume $a: \ p \in P \ p' \in P \ p \neq p'$
from this assms show $p \cap p' = \{\}$
using partition-onE subsetCE sup-ge1 by blast
qed

lemma partition-on-split2:
assumes partition-on $A \ (P \cup Q)$
shows partition-on $((\bigcup Q) \ Q)$
using assms partition-on-split1 sup-commute by metis

lemma partition-on-intersect-on-elements:
assumes partition-on $\lambda X. \ (A \cup C) \ P$
assumes $\forall X \in P. \ \exists x. \ x \in X \cap C$
shows partition-on $C \ ((\lambda X. \ X \cap C) \ ^{'P})$
proof (rule partition-onI)
fix $p$
assume $p \in (\lambda X. \ X \cap C) \ ^{'P}$
from this assms show $p \neq \{\}$ by auto
next
have $\bigcup P = A \cup C$
using assms by (auto elim: partition-onE)
from this show $\bigcup ((\lambda X. \ X \cap C) \ ^{'P}) = C \ by \ auto$
next
fix $p \ p'$
assume $p \in (\lambda X. \ X \cap C) \ ^{'P} \ p' \in (\lambda X. \ X \cap C) \ ^{'P} \ p \neq p'$
from this assms(1) show $p \cap p' = \{\}$
by (blast elim: partition-onE)
qed

lemma partition-on-insert-elements:
assumes $A \cap B = \{\}$
assumes partition-on B P
assumes \( f \in A \to_{E} P \)
shows partition-on \((A \cup B) \ ((\lambda X. X \cup \{x \in A. f x = X\}) \ ' P)\) (is partition-on - ?P)
proof (rule partition-onI)
fix X
assume \( X \in ?P \)
from this (partition-on B P) show \( X \neq \{\} \)
by (auto elim: partition-onE)
next
show \( \bigcup ?P = A \cup B \)
using (partition-on B P \( f \in A \to_{E} P \)) by (auto elim: partition-onE)
next
fix X Y
assume \( X \in ?P \ Y \in ?P \ X \neq Y \)
from \( \langle X \in ?P \rangle \) obtain \( X' \ where \ X' \colon X = X' \cup \{ x \in A. f x = X' \} \ X' \in P \) by auto
from \( \langle Y \in ?P \rangle \) obtain \( Y' \ where \ Y' \colon Y = Y' \cup \{ x \in A. f x = Y' \} \ Y' \in P \) by auto
from \( \langle X \neq Y \rangle \ X' \ Y' \ have \ X' \neq Y' \ by \ auto \)
from \( \langle X' \ Y' \ have \ X' \cap Y' = \{\} \rangle \)
using (partition-on B P) by (auto elim!: partition-onE)
from \( \langle X' \ Y' \ have \ X' \subseteq B \ Y' \subseteq B \rangle \)
from \( \langle X' \cap Y' = \{\} \rangle \ X' \ Y' \colon X' \neq Y' \ show \ X \cap Y = \{\} \)
using \( \langle A \cap B = \{\} \rangle \) by auto
qed

lemma partition-on-map:
assumes inj-on f A
assumes partition-on A P
shows partition-on \((f ' A) \ ((' \rangle f ' P)\)
proof –
\{
fix X Y
assume \( X \in P \ Y \in P \ f' \ X \neq f' Y \)
moreover from assms have \( \forall p \in P. \forall p' \in P. \ p \neq p' \longrightarrow p \cap p' = \{\} \ and \ inj-on f ((\bigcup P) \)
by (auto elim!: partition-onE)
ultimately have \( f' X \cap f' Y = \{\} \)
unfolding inj-on-def by auto (metis IntI empty-iff rev-image-eqI+)
\}
from assms this show partition-on \((f ' A) \ ((' \rangle f ' P)\)
by (auto intro!: partition-onI elim!: partition-onE)
qed

lemma set-of-partition-on-map:
assumes inj-on f A
shows \( (' \rangle (f ' P. partition-on A P) = \{P. partition-on (f ' A) P\} \)

10
proof (rule set-eqI')
fix x
assume x ∈ (\' f \') \{ P. partition-on A P \}
from this ⟨ inj-on f A ⟩ show x ∈ \{ P. partition-on (f \' A) P \}
  by (auto intro: partition-on-map)
next
fix P
assume P ∈ \{ P. partition-on (f \' A) P \}
from this have partition-on (f \' A) P by auto
from this have mem: \( \forall X. x \in P \implies x \in X \implies x \in f \' A \)
  by (auto elim!: partition-onE)
have (\' (f o the-inv-into A f) \') P = (\' (f o (the-inv-into A f)) \') P
  by (simp add: image-comp comp-image)
moreover have P = (\' (f o the-inv-into A f) \') P
proof (rule set-eqI')
fix X
assume X: X ∈ P
moreover from X mem have in-range: \( \forall x \in X. x \in f \' A \) by auto
moreover have X = (f o the-inv-into A f) \' X
proof (rule set-eqI')
fix x
assume x ∈ X
show x ∈ (f o the-inv-into A f) \' X
proof (rule image-eqI)
  from in-range ⟨ x ∈ X ⟩ assms show x = (f o the-inv-into A f) x
    by (auto simp add: f-the-inv-into-f [of f])
  from ⟨ x ∈ X ⟩ show x ∈ X by assumption
qed
next
fix x
assume x ∈ (f o the-inv-into A f) \' X
from this obtain x' where x': x' ∈ X \& x = f (the-inv-into A f x') by auto
from in-range x' have f: f (the-inv-into A f x') ∈ X
  by (subst f-the-inv-into-f [of f]) (auto intro: inj-on f A)
from x' ⟨ X ∈ P ⟩ f show x ∈ X by auto
qed
ultimately show X ∈ (\' (f o the-inv-into A f) \') P by auto
next
fix X
assume X ∈ (\' (f o the-inv-into A f) \') P
moreover
  { fix Y
    assume Y ∈ P
    from this ⟨ inj-on f A ⟩ mem have \( \forall x \in Y. f (the-inv-into A f x) = x \)
      by (auto simp add: f-the-inv-into-f)
    from this have (f o the-inv-into A f) \' Y = Y by force
  }
ultimately show X ∈ P by auto
ultimately have \( P = (\cdot) f' (\cdot) (\text{the-inv-into } A f) \cdot P \) by simp
have \( A\text{-eq}: A = \text{the-inv-into } A f \cdot f' A \) by (simp add: assms)
from \( \\text{inj-on } f A \) have inj-on \( \text{the-inv-into } A f \cdot f' A \) by (simp add: inj-on-the-inv-into)
from this have \( (\cdot) \cdot (\text{the-inv-into } A f) \cdot P \in \{P. \text{partition-on } A P\} \) by (rule image-eqI)
qed

2 Combinatorial Basics

theory Injectivity-Solver
imports
  HOL\,-\,Library.Disjoint-Sets
  HOL\,-\,Library.Monad-Syntax
  HOL\,-\,Eisbach.Eisbach
begin

2.1 Preliminaries

These lemmas shall be added to the Disjoint Set theory.

2.1.1 Injectivity and Disjoint Families

lemma inj-on-impl-disjoint-family-on-singleton:
  assumes inj-on \( f A \)
  shows disjoint-family-on \( (\lambda x. \{f x\}) A \)
using assms disjoint-family-on-def inj-on-contraD by fastforce

2.1.2 Cardinality Theorems for Set.bind

lemma card-bind:
  assumes finite \( S \)
  assumes \( \forall X \in S. \text{finite } (f X) \)
  assumes \( \text{disjoint-family-on } f S \)
  shows \( \text{card } (S \gg f) = (\sum x \in S. \text{card } (f x)) \)
proof –
  have \( \text{card } (S \gg f) = \text{card } (\bigcup (f ^ i S)) \)
    by (simp add: bind-UNION)
also have \( \text{card } (\bigcup (f ^ i S)) = (\sum x \in S. \text{card } (f x)) \)
    using assms unfolding disjoint-family-on-def
    by (subst card-Union-image) simp+
finally show \( \? \text{thesis } \).
qd
lemma card-bind-constant:
assumes finite S
assumes \( \forall X \in S. \text{finite} (f X) \)
assumes disjoint-family-on \( f S \)
assumes \( \land. \ x \in S \Rightarrow \text{card} (f x) = k \)
shows \( \text{card} (S \triangleright= f) = \text{card} S \ast k \)
using assms by (simp add: card-bind)

lemma card-bind-singleton:
assumes finite S
assumes inj-on \( f S \)
shows \( \text{card} (S \triangleright= (\lambda x. \{f x\})) = \text{card} S \)
using assms by (auto simp add: card-bind-constant inj-on-impl-disjoint-family-on-singleton)

2.2 Third Version of Injectivity Solver

Here, we provide a third version of the injectivity solver. The original first version was provided in the AFP entry ‘Spivey’s Generalized Recurrence for Bell Numbers’. From that method, I derived a second version in the AFP entry ‘Cardinality of Equivalence Relations’. At roughly the same time, Makarius improved the injectivity solver in the development version of the first AFP entry. This third version now includes the improvements of the second version and Makarius improvements to the first, and it further extends the method to handle the new cases in the cardinality proof of this AFP entry.

As the implementation of the injectivity solver only evolves in the development branch of the AFP, the submissions of the three AFP entries that employ the injectivity solver, have to create clones of the injectivity solver for the identified and needed method adjustments. Ultimately, these three clones should only remain in the stable branches of the AFP from Isabelle2016 to Isabelle2017 to work with their corresponding release versions.

In the development version, I have now consolidated the three versions here. In the next step, I will move this version of the injectivity solver in the HOL−Library.Disjoint-Sets and it will hopefully only evolve further there.

lemma disjoint-family-onI:
assumes \( \land. \ i \in I \wedge j \in I \Rightarrow i \neq j \Rightarrow (A i) \cap (A j) = \{\} \)
shows disjoint-family-on \( A I \)
using assms unfolding disjoint-family-on-def by auto

lemma disjoint-bind: \( \land S \ T \ f \ g. (\land s t. \ S \land T \ t \Rightarrow f s \cap g t = \{\}) \Rightarrow (\{s. S \} \triangleright= f) \cap (\{t. T \} \triangleright= g) = \{\} \)
by fastforce

lemma disjoint-bind': \( \land S \ T \ f \ g. (\land s t. \ S \land t \in T \Rightarrow f s \cap g t = \{\}) \Rightarrow (S \triangleright= f) \cap (T \triangleright= g) = \{\} \)
by fastforce
lemma \textit{injectivity-solver-CollectE}:
\begin{itemize}
\item assumes $a \in \{x. \ P \ x\} \land a' \in \{x. \ P' \ x\}$
\item assumes $(P \ a \land P' \ a') \Longrightarrow W$
\item shows $W$
\end{itemize}
using \texttt{assms by auto}

lemma \textit{injectivity-solver-prep-assms-Collect}:
\begin{itemize}
\item assumes $x \in \{x. \ P \ x\}$
\item shows $P \ x \land P \ x$
\end{itemize}
using \texttt{assms by simp}

lemma \textit{injectivity-solver-prep-assms}: $x \in A \Longrightarrow x \in A \land x \in A$
by \texttt{simp

lemma \textit{disjoint-terminal-singleton}: $\forall s \ t \ X \ Y. \ s \neq t \Longrightarrow (X = Y \Longrightarrow s = t) \Longrightarrow \{X\} \cap \{Y\} = \{\}$
by \texttt{auto

lemma \textit{disjoint-terminal-Collect}:
\begin{itemize}
\item assumes $s \neq t$
\item assumes $\forall x \ x'. \ S \ x \land T \ x' \Longrightarrow x = x' \Longrightarrow s = t$
\item shows $\{x. \ S \ x\} \cap \{x. \ T \ x\} = \{\}$
\end{itemize}
using \texttt{assms by auto

lemma \textit{disjoint-terminal}:
\begin{itemize}
\item $s \neq t \Longrightarrow (\forall x \ x'. \ x \in S \land x' \in T \Longrightarrow x = x' \Longrightarrow s = t) \Longrightarrow S \cap T = \{\}$
\end{itemize}
by \texttt{blast

lemma \textit{elim-singleton}:
\begin{itemize}
\item assumes $x \in \{s\} \land x' \in \{t\}$
\item obtains $x = s \land x' = t$
\end{itemize}
using \texttt{assms by blast

method \textit{injectivity-solver} \texttt{uses rule =
\begin{itemize}
\item insert method-facts,
\item use nothing in
\item ((drule injectivity-solver-prep-assms-Collect | drule injectivity-solver-prep-assms)+)?
\item rule disjoint-family-onI;
\item ((rule disjoint-bind | rule disjoint-bind\')+)?
\item (erule elim-singleton)?
\item (erule disjoint-terminal-singleton | erule disjoint-terminal-Collect | erule disjoint-terminal);
\item (elim injectivity-solver-CollectE)?
\item rule rule;
\item assumption+
\end{itemize}
)

end
3 Cardinality of Set Partitions

theory Card-Partitions
imports
  HOL−Library.Stirling
  Set-Partition
  Injectivity-Solver
begin

lemma set-partition-on-insert-with-fixed-card-eq:
  assumes finite A
  assumes a /∈ A
  shows \{ P. partition-on (insert a A) P ∧ card P = Suc k \} = (do { 
    P <- \{ P. partition-on A P ∧ card P = Suc k \}; 
    p <- P; 
    \{ insert (insert a p) (P - \{ p \}) \} 
  }) ∪ (do { 
    P <- \{ P. partition-on A P ∧ card P = k \}; 
    \{ insert \{ a \} P \} 
  }) (is {?S = {?T})
proof
  show {?S ⊆ {?T
    proof
      fix P
      assume P ∈ \{ P. partition-on (insert a A) P ∧ card P = Suc k \}
      from this have partition-on (insert a A) P and card P = Suc k by auto
      show P ∈ {?T
        proof cases
        assume \{ a \} ∈ P
        have partition-on A (P - \{ \{ a \} \})
          using \{ a \} ∈ P \ ⟨ partition-on (insert a A) P \ ⟨ THEN partition-on-Diff, of \{ \{ a \} \} \ ⟨ a /∈ A \⟩ by auto
        moreover from \{ a \} ∈ P \ ⟨ card P = Suc k \⟩ have card (P - \{ \{ a \} \}) = k
          by (subst card-Diff-singleton) (auto intro: card-le-0-finite)
        moreover from \{ a \} ∈ P \ ⟨ have P = insert \{ a \} (P - \{ \{ a \} \}) by auto
        ultimately have P ∈ \{ P. partition-on A P ∧ card P = k \} ⇔ (\lambda P. \{ insert \{ a \} P \})
          by auto
        from this show ?thesis by auto
      next
        assume \{ a \} /∈ P
        let ?p' = (THE X. a ∈ X ∧ X ∈ P)
        let ?p = (THE X. a ∈ X ∧ X ∈ P) - \{ a \}
        let ?P' = insert ?p (P - \{ ?p' \})
        from \langle partition-on (insert a A) P \⟩ have a ∈ (THE X. a ∈ X ∧ X ∈ P)
          using partition-on-in-the-unique-part by fastforce
        from \langle partition-on (insert a A) P \⟩ have (THE X. a ∈ X ∧ X ∈ P) ∈ P
using partition-on-the-part-mem by fastforce
from this (\(\text{partition-on (insert a A)} \ P\)) have (\(\text{THE X. a } \in \text{ X } \land \text{ X } \in P\)) – \{a\} \not\in P
  using partition-subset-imp-notin \(\{a \in (\text{THE X. a } \in \text{ X } \land \text{ X } \in P)\}\) by blast
  have (\(\text{THE X. a } \in \text{ X } \land \text{ X } \in P\)) \not\in \{a\}
  using (\(\text{THE X. a } \in \text{ X } \land \text{ X } \in P\)) \(\{a\} \not\in P\) by auto
from (\(\text{partition-on (insert a A)} \ P\)) have (\(\text{THE X. a } \in \text{ X } \land \text{ X } \in P\)) \subseteq \text{ insert a A}
  using (\(\text{THE X. a } \in \text{ X } \land \text{ X } \in P\)) \(\{a\} \not\in P\) \(\text{ by auto}
from \(\text{partition-on (insert a A)} \ P\) \(\text{finite A}\) have finite P
by (meson finite.insert1 finite-elements)
from \(\text{partition-on (insert a A)} \ P\) \(\{a \notin A\}\) have partition-on \(\text{(A } \not\in \text{?p)} \ (P - \{\text{?p}\})\)
using facts-on-the-part-of by (auto intro: partition-on-remove-singleton)
from this have partition-on \(\text{A } \not\in \text{?P}'\)
  using facts-on-the-part-of by (auto intro: partition-on-insert simp add: disjoint-iff)
moreover have \(\text{card } \not\in \text{?P}' = \text{Suc k}\)
proof –
  from \(\text{card P = Suc k}\) have \(\text{card (P } - \{\text{THE X. a } \in \text{ X } \land \text{ X } \in P\}) = k\)
  using \(\text{finite P}\) \(\text{(THE X. a } \in \text{ X } \land \text{ X } \in P) \in P\) by simp
  from this show \(\text{?thesis}\)
  using \(\text{finite P}\) \(\text{(THE X. a } \in \text{ X } \land \text{ X } \in P) - \{a\} \not\in P\) by (simp add: card-insert-if)
qed
moreover have \(\text{?p } \in \text{?P}'\) by auto
moreover have \(\text{P } = \text{insert (insert a ?p)} \ (\text{?P}' - \{\text{?p}\})\)
  using facts-on-the-part-of by (auto simp add: insert-absorb)
ultimately have \(\text{P } \in \{\text{P. partition-on A P } \land \text{ card P } = \text{Suc k}\} \Longrightarrow (\lambda P. P\)
  \(\Longrightarrow (\lambda P. \{\text{insert (insert a p)} \ (P - \{p\})\})\)
  by auto
  then show \(\text{?thesis}\) by auto
qed
next
show \(\text{?T } \subseteq \text{?S}\)
proof
fix \(P\)
assume \(P \in \text{?T}\) (is - \(\text{?subexpr1} \cup \text{?subexpr2}\))
from this show \(P \in \text{?S}\)
proof
  assume \(P \in \text{?subexpr1}\)
  from this obtain \(p \text{ P}'\) where \(P = \text{insert (insert a p)} \ (\text{P}' - \{p\})\)
  and partition-on \(\text{A P}'\) and \(\text{card P}' = \text{Suc k}\) and \(\text{p } \in \text{P}'\) by auto
from \(\text{p } \in \text{P}'\) (partition-on \(\text{A P}'\)) have partition-on \(\text{(A } \not\in \text{p)} \ (\text{P}' - \{p\})\)
by (simp add: partition-on-remove-singleton)
from partition-on A \( P \) \( \text{finite} \) A \ have finite \( P \)
  using \( \{ P = \} \ \text{finite-elements} \) by auto
from partition-on A \( P \) \( \text{fin} \) \( a \notin A \) \ have insert a \( p \notin P \') \( \{ p \} \)
  using partition-onD1 by fastforce
from |\( P = \) | this \( \langle \text{card} \ P' \rangle \) \( \text{Suc} \ k \) \( \langle \text{finite} \ P \rangle \) \( \{ p \in P \}' \)
have card P = Suc k by auto
moreover have partition-on (insert a A) \( P \)
  using \( \langle \text{partition-on} \ (A - p) \ P' \rangle \) \( \{ a \notin A \} \) \( \{ p \in P \}' \)
by (auto intro!: partition-on-insert dest: partition-onD1 simp add: disjoint-if)
ultimately show P \( \in \ ?S \) by auto
next
assume P \( \in \ ?\text{subexpr2} \)
from this obtain P' \ where P = insert \( \{ a \} \) P' \ and partition-on A \( P' \) \ and
  card P' = k by auto
from partition-on A \( P \) \( \text{fin} \) \( a \notin A \) \ have \( \{ P = \} \ \text{finite-elements} \) by auto
from partition-on A \( P \) \( \text{fin} \) \( a \notin A \) \ have \( \{ a \} \notin P' \)
  using partition-onD1 by fastforce
from |\( P = \) | insert \( \{ a \} \) P' \( \langle \text{card} \ P' \rangle \) \( \text{Suc} \ k \) \( \langle \text{finite} \ P \rangle \) \( \langle \text{finite} \ P' \rangle \)
by auto
moreover from partition-on A \( P \) \( \text{fin} \) \( a \notin A \) \ have partition-on (insert a A) \( P \)
  using \( \langle P = \text{insert} \{ a \} \ P' \rangle \) by (simp add: partition-on-insert-singleton)
ultimately show P \( \in \ ?S \) by auto
qed

lemma injectivity-subexpr1:
  assumes a \( \notin A \)
  assumes X \( \in P \wedge X' \in P' \)
  assumes \( \langle \text{insert} \ (\text{insert} a X) \ (P - \{ X \}) \rangle = \langle \text{insert} \ (\text{insert} a X') \ (P' - \{ X' \}) \rangle \)
  assumes \( \langle \text{partition-on} \ A \ P \wedge \text{card} P = \text{Suc} k \rangle \wedge \langle \text{partition-on} \ A \ P' \wedge \text{card} P' = \text{Suc} k \rangle \)
  shows P = P' \ and X = X'
proof
  from assms(1, 2, 4) have a \( \notin X \) a \( \notin X' \)
    using partition-onD1 by auto
from assms(1, 4) have insert a X \( \notin P \) insert a X' \( \notin P' \)
  using partition-onD1 by auto
from assms(1, 3, 4) have insert a X = insert a X'
    by (metis Diff_iff insertE insert1 mem_simps(9) partition-onD1)
from this \( \langle a \notin X \rangle \langle a \notin X' \rangle \) show X = X'
  by (meson insert-ident)
from assms(2, 3) show P = P' 
  using \( \langle \text{insert} \ a X = \langle \text{insert} a X \rangle \ P \rangle \langle \text{insert} \ a X' \notin P' \rangle \langle \text{insert} a X' \notin P' \rangle \)
  by (metis insert-Diff insert-absorb insert-commute insert-ident)
lemma injectivity-subexpr2:
  assumes a /∈ A
  assumes insert {a} P = insert {a} P'
  assumes (partition-on A P ∧ card P = k') ∧ partition-on A P' ∧ card P' = k'
  shows P = P'
proof
  from assms(1, 3) have {a} /∈ P and {a} /∈ P'
    using partition-onD1 by auto
  from ⟨{a} /∈ P⟩ have P = insert {a} P − {{a}} by simp
  also from ⟨insert {a} P = insert {a} P'⟩ have … = insert {a} P' − {{a}} by simp
  finally show ?thesis.
qed

theorem card-partition-on:
  assumes finite A
  shows card {P. partition-on A P ∧ card P = k} = Stirling (card A) k
using assms
proof (induct A arbitrary: k)
case empty
  have eq: {P. P = {} ∧ card P = 0} = {{}} by auto
  show ?case by (cases k) (auto simp add: partition-on-empty eq)
next
case (insert a A)
  from this show ?case
proof (cases k)
  case 0
  from insert(1) have empty: {P. partition-on (insert a A) P ∧ card P = 0} = {}
    unfolding partition-on-def by (auto simp add: card-eq-0-iff finite-UnionD)
  from 0 insert show ?thesis by (auto simp add: empty)
next
case (Suc k')
let ?subexpr1 = do
  P <- {P. partition-on A P ∧ card P = Suc k'};
  p <- P;
  {insert (insert a p) (P − {p})}

let ?subexpr2 = do
  P <- {P. partition-on A P ∧ card P = k'};
  {insert {a} P}

let ?expr = ?subexpr1 ∪ ?subexpr2
have card {P. partition-on (insert a A) P ∧ card P = k} = card ?expr
  using ⟨finite A; a /∈ A; k = Suc k'⟩ by (simp add: set-partition-on-insert-with-fixed-card-eq)
also have card ?expr = Stirling (card A) k' + Stirling (card A) (Suc k') * Suc
\begin{proof}
\begin{enumerate}
\item \textbf{proof} --
\begin{enumerate}
\item \textbf{have} finite ?subexpr1 \land card ?subexpr1 = Stirling (card A) (Suc k') \land Suc k'
\begin{proof} --
\item from \{finite A\} have finite \{P. partition-on A \land card P = Suc k'\}
\item have finite \{P. partition-on A \land card P = Suc k'\}
\item more have \(\forall X \in \{P. partition-on A \land card P = Suc k'\}. \text{finite}(X) \supseteq (\lambda p. \{\text{insert}(\text{insert} a p)(X - \{p\})\})\)
\item using finite-elements (finite A) finite-bind
\item have \{finite ?subexpr1\}
\item have \{finite ?subexpr2\}
\item more have disjoint-family-on \{\(\lambda p. \{\text{insert}(\text{insert} a p)(P - \{p\})\}\)\} (\(\lambda p. \{\text{insert}(\text{insert} a p)(P - \{p\})\}\)
\item eventually show ?thesis by \{simp add: card-bind-singleton\}
\item ultimately have card ?subexpr1 = card \{P. partition-on A \land card P = Suc k'\} \land Suc k'
\item by (subst card-bind-singleton) simp+
\item have \{finite ?subexpr1\}
\item have \{finite \{P. partition-on A \land card P = Suc k'\}\}
\item \(\forall X \in \{P. partition-on A \land card P = Suc k'\}. \text{finite}(X) \supseteq (\lambda p. \{\text{insert}(\text{insert} a p)(X - \{p\})\})\)
\item by (auto intro: finite-bind)
\item ultimately show ?thesis by blast
\item finally have card ?subexpr2 = Stirling (card A) k'
\item by (simp add: card-bind-singleton)
\item also have \(\ldots = \text{Stirling}(\text{card} A) k'\)
\item using insert.hyps(3)
\item finally have card ?subexpr2 = Stirling (card A) k'
\item more have finite ?subexpr2
\end{enumerate}
\end{proof}
\end{enumerate}
\end{proof}
ultimately show ?thesis by blast

qed

moreover have ?subexpr1 ∩ ?subexpr2 = {}

proof
  have ∀ P ∈ ?subexpr1. {a} ∉ P
    using insert.hyps(2) by (force elim!: partition-onE)
  moreover have ∀ P ∈ ?subexpr2. {a} ∈ P by auto
  ultimately show ?subexpr1 ∩ ?subexpr2 = {} by blast
  qed
ultimately show ?thesis
  by (simp add: card-Un-disjoint)
  qed

also have . . . = Stirling (card (insert a A)) k
  using insert(1, 2) (k = Suc k′) by simp
finally show ?thesis .
  qed

thesis card-partition-on-at-most-size:
  assumes finite A
  shows card \{P. partition-on A P ∧ card P ≤ k\} = (∑j≤k. Stirling (card A) j)
proof —
  have card \{P. partition-on A P ∧ card P ≤ k\} = card (∪j≤k. \{P. partition-on A P ∧ card P = j\})
    by (rule arg-cong[where f=card]) auto
  also have . . . = (∑j≤k. card \{P. partition-on A P ∧ card P = j\})
    by (subst card-UN-disjoint) (auto simp add: finite A; finitely-many-partition-on)
  also have (∑j≤k. card \{P. partition-on A P ∧ card P = j\}) = (∑j≤k. Stirling (card A) j)
    using ⟨finite A⟩ by (simp add: card-partition-on)
  finally show ?thesis .
  qed

thesis partition-on-size1:
  assumes finite A
  shows \{P. partition-on A P ∧ (∀ X∈P. card X = 1)\} = (\{λa. \{a\} \hole A\}
proof
  show \{P. partition-on A P ∧ (∀ X∈P. card X = 1)\} ⊆ (\{λa. \{a\} \hole A\}
proof
fix P
  assume P: P ∈ \{P. partition-on A P ∧ (∀ X∈P. card X = 1)\}
  have P = (λa. \{a\} \hole A
proof
  show P ⊆ (λa. \{a\} \hole A
proof
  fix X
  assume X ∈ P
  from P this obtain x where X = \{x\}

20
by (auto simp add: card-Suc-eq)
from this \(\langle X \in P, \; \text{have} \; x \in A \rangle\)
    using \(P\) unfolding partition-on-def by blast
from this \(\langle X = \{x\}, \; \text{show} \; x \in (\lambda a. \{a\}) \cdot A \rangle\) by auto
qed
next
show \((\lambda a. \{a\}) \cdot A \subseteq P\)
proof
  fix \(X\)
  assume \(X \in (\lambda a. \{a\}) \cdot A\)
  from this obtain \(x\) where \(X = \{x\} \cdot x \in A\) by auto
  have \(\bigcup P = A\)
    using \(P\) unfolding partition-on-def by blast
  from this \(\langle x \in A, \; \text{obtain} \; X' \cdot x \in X' \rangle\) and \(X' \in P\)
    using card-1-singletonE by blast
from this \(X(1) : X' \in P; \; \text{show} \; X \in P\) by auto
qed
qed
from this show \(P \in \{(\lambda a. \{a\}) \cdot A\}\) by auto
qed
next
show \(\{(\lambda a. \{a\}) \cdot A\} \subseteq \{P. \; \text{partition-on} \; A \cdot P \wedge (\forall X \in P. \; \text{card} \; X = 1)\}\)
proof
  fix \(P\)
  assume \(P \in \{(\lambda a. \{a\}) \cdot A\}\)
  from this have \(P: \; P = (\lambda a. \{a\}) \cdot A\) by auto
  from this have partition-on \(A \cdot P\) by (auto intro: partition-onI)
from this \(\langle P \rangle : P \in \{P. \; \text{partition-on} \; A \cdot P \wedge (\forall X \in P. \; \text{card} \; X = 1)\}\) by auto
qed
qed

theorem card-partition-on-size1:
assumes \(\text{finite} \; A\)
shows \(\text{card} \; \{P. \; \text{partition-on} \; A \cdot P \wedge (\forall X \in P. \; \text{card} \; X = 1)\} = 1\)
using assms partition-on-size1 by fastforce

lemma card-partition-on-size1-eq-1:
assumes \(\text{finite} \; A\)
assumes \(\text{card} \; A \leq k\)
shows \(\text{card} \; \{P. \; \text{partition-on} \; A \cdot P \wedge \text{card} \; P \leq k \wedge (\forall X \in P. \; \text{card} \; X = 1)\} = 1\)
proof
  {  
    fix \(P\)
    assume \(\text{partition-on} \; A \cdot P \forall X \in P. \; \text{card} \; X = 1\)
from this have \(P \in \{P. \; \text{partition-on} \; A \cdot P \wedge (\forall X \in P. \; \text{card} \; X = 1)\}\) by simp
from this have \( P \in \{(\lambda a. \{a\}) \cdot A\} \)
using partition-on-size1 ⟨finite A⟩ by auto
from this have \( P = (\lambda a. \{a\}) \cdot A \) by auto
moreover from this have \( \text{card } P = \text{card } A \)
by (auto intro: card-image)

from this have \( \{P. \text{partition-on } A P \land \text{card } P \leq k \land (\forall X \in P. \text{card } X = 1)\} = \{P. \text{partition-on } A P \land (\forall X \in P. \text{card } X = 1)\} \)
using ⟨\text{card } A \leq k⟩ by auto
from this show ?thesis
using ⟨finite A⟩ by (simp only: card-partition-on-size1)
qed

lemma card-partition-on-size1-eq-0:
assumes finite A
assumes \( k < \text{card } A \)
sows \( \text{card } \{P. \text{partition-on } A P \land \text{card } P \leq k \land (\forall X \in P. \text{card } X = 1)\} = 0 \)
proof –
{ \fix P
assume partition-on A P \forall X \in P. \text{card } X = 1
from this have \( P \in \{P. \text{partition-on } A P \land (\forall X \in P. \text{card } X = 1)\} \) by simp
from this have \( P \in \{(\lambda a. \{a\}) \cdot A\} \)
using partition-on-size1 ⟨finite A⟩ by auto
from this have \( P = (\lambda a. \{a\}) \cdot A \) by auto
from this have \( \text{card } P = \text{card } A \)
by (auto intro: card-image)
}
from this assms(2) have \( \{P. \text{partition-on } A P \land \text{card } P \leq k \land (\forall X \in P. \text{card } X = 1)\} = {} \)
using Collect-empty-eq leD by fastforce
from this show ?thesis by (simp only: card-empty)
qed

end

References


numbers_of_the_second_kind&oldid=693800357, [Online; accessed 12-December-2015].