Cardinality of Set Partitions

Lukas Bulwahn

February 23, 2021

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
   1.1 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 12
   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
HOL−Library.FuncSet

begin

1.1 Useful Additions to Main Theories

lemma set-eqI:
  assumes ∀x. x ∈ A ⇒ x ∈ B
  assumes ∀x. x ∈ B ⇒ x ∈ A
  shows A = B
using assms by auto

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

1.2 Introduction and Elimination Rules

The definition of partition-on is in HOL−Library.Disjoint-Sets.

lemma partition-onI:
  assumes ∀p. p ∈ P ⇒ p ≠ ∅
  assumes P = {p ∈ P}
  assumes ∀p p'. p ∈ P ⇒ p' ∈ P ⇒ p ≠ p' ⇒ p ∩ p' = {}
  shows partition-on A P
using assms unfolding partition-on-def disjoint-def by blast

lemma partition-onE:
  assumes partition-on A P
  obtains ∀p. p ∈ P ⇒ p ≠ ∅
  {P = {p ∈ P}
  {∀p p'. p ∈ P ⇒ p' ∈ P ⇒ p ≠ p' ⇒ p ∩ p' = {}}
using assms unfolding partition-on-def disjoint-def by blast

1.3 Basic Facts on Set Partitions

lemma partition-onD4: partition-on A P \( p \in P \implies q \in P \implies x \in p \implies x \in q \)
  by (auto simp: partition-on-def disjoint-def)

lemma partition-subset-imp-notin:
  assumes partition-on A P X ∈ P
  assumes X' ⊂ X
  shows X' ∉ P
proof
  assume X' ∈ P
  from \( X' \in P \) \( partition-on A P \) have X' ≠ {} using partition-onD3 by blast
moreover from \( X' \in P \) \( X \in P \) \( partition-on A P \) \( X' \subset X \) have disjoint X X'
  by (metis disjoint_def disjointD inf.strict-order-iff partition-onD2)
moreover note \( X' \subset X \) 
ultimately show False
by (meson all-not-in-conv disjnt-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 disjnt-def)

lemma partition-on-UN:
assumes A: partition-on A B and B: \( \bigwedge b. \ b \in B \implies partition-on b (P b) \)
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: \( p \in P \ i \in B \) and j: \( q \in P \ j \in B \)
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: disjntD)
next
assume i \neq j
then have disjnt i j
using i j A[THEN partition-onD2, of i, symmetric] B[THEN partition-onD1, of i, symmetric] B[THEN partition-onD1, of j, symmetric] i j by auto
ultimately show ?thesis
by (auto simp: disjnt-def)
qed

lemma partition-on-notemptyI:
assumes partition-on A P
assumes A \neq \{\}
shows P \neq \{\}
using asms 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 $A$ $Q$
  assumes partition-on $B$ $Q$
  shows $A = B$
using assms by (fastforce elim: partition-onE)

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

An alternative formulation of $\exists \{\text{partition-on} \ ?A \ ?P; \ \text{disjoint} \ ?A \ ?X; \ ?X \neq \{\}; \ ?A \cup ?X = ?A'\} \Rightarrow \text{partition-on} \ ?A' \ (\text{insert} \ ?X \ ?P)$

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

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

**lemma** partition-on-remove-singleton:
  assumes partition-on $\ A \ ?P \ X \ \in \ P \ A - X = A'$
  shows partition-on $\ A' \ (P - \{X\})$
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 $\ A \ P$
  assumes $x \in A$
  shows $\exists \! X. \ x \in X \ \land \ X \in P$
proof –
from ⟨partition-on A P⟩ have ∪ P = A
  by (auto elim: partition-onE)
from this ⟨x ∈ A⟩ obtain X where X: x ∈ X ∧ X ∈ P by blast
{
  fix Y
  assume x ∈ Y ∧ Y ∈ 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 ∈ A
  shows (THE X. x ∈ X ∧ X ∈ P) ∈ P
proof –
  from ⟨x ∈ A⟩ have ∃!X. x ∈ X ∧ X ∈ P
    using ⟨partition-on A P⟩ by (simp add: partition-on-partition-on-unique)
  from this show (THE X. x ∈ X ∧ X ∈ P) ∈ P
    by (metis (no-types, lifting) the1)
qed

lemma partition-on-in-the-unique-part:
  assumes partition-on A P
  assumes x ∈ A
  shows x ∈ (THE X. x ∈ X ∧ X ∈ P)
proof –
  from assms have ∃!X. x ∈ X ∧ X ∈ P
    by (simp add: partition-on-partition-on-unique)
  from this show ?thesis
    by (metis (mono-tags, lifting) the1′)
qed

lemma partition-on-the-part-eq:
  assumes partition-on A P
  assumes x ∈ X X ∈ P
  shows (THE X. x ∈ X ∧ X ∈ P) = X
proof –
  from ⟨x ∈ X⟩ ⟨X ∈ P⟩ have x ∈ A
    using ⟨partition-on A P⟩ by (auto elim: partition-onE)
  from this have ∃!X. x ∈ X ∧ X ∈ P
    using ⟨partition-on A P⟩ by (simp add: partition-on-partition-on-unique)
  from ⟨x ∈ X⟩ ⟨X ∈ P⟩ this show (THE X. x ∈ X ∧ X ∈ P) = X
    by (auto intro: the1-equality)
qed

lemma the-unique-part-alternative-def:
assumes partition-on A P
assumes x ∈ A
shows (THE X. x ∈ X ∧ X ∈ P) = {y. ∃X ∈ P. x ∈ X ∧ y ∈ X}
proof
  show (THE X. x ∈ X ∧ X ∈ P) ⊆ {y. ∃X ∈ P. x ∈ X ∧ y ∈ X}
  proof
    fix y
    assume y ∈ (THE X. x ∈ X ∧ X ∈ P)
    moreover from ⟨x ∈ A⟩ have x ∈ {THE X. x ∈ X ∧ X ∈ P}
      using ⟨partition-on A P⟩ partition-on-in-the-unique-part by force
    moreover from ⟨x ∈ A⟩ have {THE X. x ∈ X ∧ X ∈ P} ∈ P
      using ⟨partition-on A P⟩ partition-on-the-part-eq by force
    ultimately show y ∈ {y. ∃X ∈ P. x ∈ X ∧ y ∈ X} by auto
  qed
next
  show {y. ∃X ∈ P. x ∈ X ∧ y ∈ X} ⊆ (THE X. x ∈ X ∧ X ∈ P)
  proof
    fix y
    assume y ∈ {y. ∃X ∈ P. x ∈ X ∧ y ∈ X}
    from this obtain X where x ∈ X and y ∈ X and X ∈ P by auto
    from ⟨x ∈ X⟩ ⟨X ∈ P⟩ have {THE X. x ∈ X ∧ X ∈ P} = X
      using ⟨partition-on A P⟩ partition-on-the-part-eq by force
    from this ⟨y ∈ X⟩ show y ∈ {THE X. x ∈ X ∧ X ∈ P} by simp
  qed

lemma partition-on-all-in-part-eq-part:
  assumes partition-on A P
  assumes X' ∈ P
  shows {x ∈ A. (THE X. x ∈ X ∧ X ∈ P) = X'} = X'
  proof
    show {x ∈ A. (THE X. x ∈ X ∧ X ∈ P) = X'} ⊆ X'
      using assms(1) partition-on-in-the-unique-part by force
next
  show X' ⊆ {x ∈ A. (THE X. x ∈ X ∧ X ∈ P) = X'}
  proof
    fix x
    assume x ∈ X'
    from ⟨x ∈ X'⟩ ⟨X' ∈ P⟩ have x ∈ A
      using ⟨partition-on A P⟩ by (auto elim: partition-onE)
    moreover from ⟨x ∈ X'⟩ ⟨X' ∈ P⟩ have {THE X. x ∈ X ∧ X ∈ P} = X'
      using ⟨partition-on A P⟩ partition-on-the-part-eq by fastforce
    ultimately show x ∈ {x ∈ A. (THE X. x ∈ X ∧ X ∈ P) = X'} by auto
  qed

lemma partition-on-part-characteristic:
  assumes partition-on A P
  assumes X ∈ P x ∈ 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 `partition-on A P. partition-onE` by blast
  from $(x \in X) \land (X \in P)$ have $X = (\{x. x \in X \land x \in P\})$
    using `partition-on A P` by (simp add: partition-on-the-part-eq)
  also from $(x \in A)$ have $(\{x. x \in X \land y \in X\}) = \{ y. \exists X \in P. x \in X \land y \in X \}$
    using `partition-on A P` theorem-part-alternative-def by force
  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 (simp add: `partition-on-empty`)
  next
    case `insert a A`
    show `?case` proof (cases `{a} \in P`)
      case `True`
        have `prop-partition-on: \forall p \in P. p \neq {a} \cup P = \text{insert } a A`
          \forall p \in P. \forall p' \in P. p \neq p' \implies p \cap p' = {}
        using `partition-on (insert a A) P` by (fastforce elim: partition-onE)+
        from this(2, 3) `{a} \notin A \land {a} \in P` have `A-eq: A = (\{P - \{a\}\})`
          by auto (metis Int-iff UnionI empty-iff insert-iff)
        from `prop-partition-on A-eq` `partition-on: partition-on A (P - \{a\})`
          by (intro `partition-onI`) 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 `finite A` partition-on] by simp`
    next
      case `False`
        from `partition-on (insert a A) P` obtain `p` where `p-def: p \in P \land a \in p`
          by (blast elim: partition-onE)
        from `partition-on (insert a A) P` `p-def` `a-notmem: \forall p' \in P - \{p\}. a \notin p'`
by (blast elim: partition-onE)
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)
  from p-def this ⟨a /∈ A⟩ a-notmem show ⋃(insert (p - {a}) (P - {p})) = A by auto
  next
  show ⋂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 p-def (finite 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
  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 \((A \cup C)\) \(P\)
  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 \(\bigcup P = A \cup C\) by auto
    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 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 $f \in A \rightarrow E \; P$
shows partition-on $(A \cup B) \; ((\lambda X. X \cup \{x \in A. \; f \; x = X\}) \; P) \; (\text{is partition-on} \;$ $\; \; ?P)$

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 $B \; P$; $f \in A \rightarrow E \; P$) by (auto elim: partition-onE)

next
  fix $X \; Y$
  assume $X \in \; \; ?P \; Y \in \; \; ?P \; X \neq Y$
  from $X \in \; \; ?P$ obtain $X'$ where $X' \; : \; X' = X' \cup \{x \in A. \; f \; x = X'\} \; X' \in P$
  by auto
  from $Y \in \; \; ?P$ obtain $Y'$ where $Y' \; : \; Y' = Y' \cup \{x \in A. \; f \; x = Y'\} \; Y' \in P$
  by auto
  from $X \neq Y \; \; X' \neq Y'$ by auto
  from this $X' \cap Y' = \{\}$
  using (partition-on $B \; P$) by (auto elim!: partition-onE)
  from $X' \subseteq B \; Y' \subseteq B$
  using (partition-on $B \; P$) by (auto elim!: partition-onE)
  from this $X' \cap Y' = \{\}$ $X' \; \; Y'$ have $X' \neq Y'$
  show $X \cap Y = \{\}$
  using $A \cap B = \{\}$ by auto
qed

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

lemma set-of-partition-on-map:
  assumes inj-on $f \; A$

shows \((\forall x \in (\\cdot (f)) \cdot \{P. \text{partition-on } A f\}) \cdot P\)

**proof** (rule set-eqI)

fix \(x\)

assume \(x \in (\\cdot (f)) \cdot \{P. \text{partition-on } A f\}\)

from this \((\text{inj-on } f A)\) show \(x \in (\\cdot (f)) \cdot \{P. \text{partition-on } (f' \cdot A) \cdot P\}\)

by (auto intro: partition-on-map)

next

fix \(P\)

assume \(P \in (\\cdot (f)) \cdot \{P. \text{partition-on } (f' \cdot A) \cdot P\}\)

from this have \(\text{partition-on } (f' \cdot A) \cdot P\) by auto

from this have \(\text{mem}: \forall \ x. \ X \in P \Longrightarrow x \in X \Longrightarrow x \in f' \cdot A\)

by (auto elim!: partition-onE)

have \((\forall x \in (\\cdot (f' \cdot A)) \cdot P = (\forall x \in (\\cdot (f' \cdot (\text{the-inv-into } A f)) \cdot P\)\)

by (simp add: image-image cong: image-cong-simp)

moreover have \(P = (\forall x \in (\\cdot (f' \cdot \text{the-inv-into } A f)) \cdot P\)

**proof** (rule set-eqI)

fix \(X\)

moreover from \(X\) have \(\text{in-range}: \forall x \in X. \ x \in f' \cdot A\) by auto

moreover have \(X = (f' \cdot \text{the-inv-into } A f) \cdot X\)

**proof** (rule set-eqI)

fix \(x\)

assume \(x \in X\)

show \(x \in (f' \cdot \text{the-inv-into } A f) \cdot X\)

**proof** (rule image-eqI)

from \(\text{in-range}: \forall x \in X\) have \(x \in (f' \cdot \text{the-inv-into } A f) \cdot x\)

by (auto simp add: f-the-inv-into-f[of f])

from \(x \in X\) show \(x \in X\) by assumption

qed

next

fix \(x\)

assume \(x \in (f' \cdot \text{the-inv-into } A f) \cdot X\)

from this obtain \(x' \cdot \text{where}\) \(x' \cdot x' \in X \land x = f' (\text{the-inv-into } A f \cdot x')\) by auto

from \(\text{in-range}\) \(x' \cdot \text{have}\) \(f: f' (\text{the-inv-into } A f \cdot x') \in X\)

by (subst f-the-inv-into-f[of f]) (auto intro: inj-on f A)

from \(x' \cdot X \in P\) show \(x \in X\) by auto

qed

ultimately show \((\forall x \in (\\cdot (f' \cdot \text{the-inv-into } A f)) \cdot P\) by auto

next

fix \(X\)

assume \(X \in (\\cdot (f' \cdot \text{the-inv-into } A f)) \cdot P\)

moreover

\{

fix \(Y\)

assume \(Y \in P\)

from this (inj-on f A) have \(\forall x \in Y. \ f (\text{the-inv-into } A f \cdot x) = x\)

by (auto simp add: f-the-inv-into-f)

from this have \((f' \cdot \text{the-inv-into } A f) \cdot Y = Y\) by force

\}

11
ultimately show \( X \in P \) by \texttt{auto}

qed

ultimately have \( P = (\cdot) f \cdot (\cdot) \) \((\text{the-inv-into } A f) \cdot P\) by \texttt{simp}

have \( A\text{-eq: } A = \text{the-inv-into } A f \cdot f \cdot A\) by \texttt{(simp add: assms)}

from \((\text{inj-on } f A)\) have \( \text{inj-on} \ (\text{the-inv-into } A f) \ (f \cdot A)\)

using \((\text{partition-on} (f \cdot A) P)\) by \texttt{(simp add: inj-on-the-inv-into)}

from this have \((\cdot) \ (\text{the-inv-into } A f) \cdot P \in \{P. \text{partition-on } A P\}\)

using \((\text{partition-on} (f \cdot A) P)\) by \texttt{(subst A-eq, auto intro!: partition-on-map)}

from \(P\) this show \(P \in (\cdot) ((\cdot) f) \cdot \{P. \text{partition-on } A P\}\) by \texttt{(rule image-eqI)}

qed

end

2 Combinatorial Basics

theory \textit{Injectivity-Solver}

imports
\(\text{HOL-Library.Disjoint-Sets}\)
\(\text{HOL-Library.Monad-Syntax}\)
\(\text{HOL-Eisbach.Eisbach}\)

begin

2.1 Preliminaries

These lemmas shall be added to the Disjoint Set theory.

2.1.1 Injectivity and Disjoint Families

lemma \textit{inj-on-impl-disjoint-family-on-singleton}:

assumes \(\text{inj-on } f A\)

shows \(\text{disjoint-family-on} (\lambda x. \{f x\}) A\)

using \(\text{assms disjoint-family-on-def inj-on-contraD}\) by \texttt{fastforce}

2.1.2 Cardinality Theorems for \textit{Set.bind}

lemma \textit{card-bind}:

assumes \(\text{finite } S\)

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 \(\text{card } (S \gg f) = \text{card } (\bigcup (f \cdot S))\)

by \texttt{(simp add: bind-UNION)}

also have \(\text{card } (\bigcup (f \cdot S)) = (\sum x \in S. \text{card } (f x))\)

using \(\text{assms unfolding disjoint-family-on-def}\) by \texttt{(simp add: card-UN-disjoint)}

finally show \(\text{thesis}\).

qed
lemma card-bind-constant:
  assumes finite S
  assumes ∀ X ∈ S. finite (f X)
  assumes disjoint-family-on f S
  assumes ∃ x ∈ S. card (f x) = k
  shows card (S ⇒ f) = card S * k
  using assms by (simp add: card-bind)

lemma card-bind-singleton:
  assumes finite S
  assumes inj-on f S
  shows card (S ⇒ (λ x. {f x})) = 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 ∀ i j. i ∈ I ∧ j ∈ I ⇒ i ≠ j ⇒ (A i) ∩ (A j) = {}
  shows disjoint-family-on A I
  using assms unfolding disjoint-family-on-def by auto

lemma disjoint-bind: ∀ S T f g. (∀ s t. S s ∧ T t ⇒ f s ∩ g t = {} ) ⇒ (∃ s. S s ⇒ f) ∩ (∃ t. T t ⇒ g) = {}
  by fastforce

lemma disjoint-bind′: ∀ S T f g. (∀ s t ∈ S ∧ t ∈ T ⇒ f s ∩ g t = {} ) ⇒ (S ⇒ f) ∩ (T ⇒ g) = {}
  by fastforce
lemma injectivity-solver-CollectE:
  assumes \( a \in \{ x. \, P \, x \} \land a' \in \{ x. \, P' \, x \} \)
  assumes \(( P \, a \land P' \, a') \implies W \)
  shows \( W \)
using \, \text{assms by} \, \text{auto}

lemma injectivity-solver-prep-assms-Collect:
  assumes \( x \in \{ x. \, P \, x \} \)
  shows \( P \, x \land P' \, x \)
using \, \text{assms by} \, \text{simp}

lemma injectivity-solver-prep-assms:
  \( x \in A \implies x \in A \land x \in A \)
by \, \text{simp}

lemma disjoint-terminal-singleton: \( \forall \, s \, t \, X \, Y. \, s \neq t \implies (X = Y \implies s = t) \implies \{ X \} \cap \{ Y \} = \{ \} \)
by \, \text{auto}

lemma disjoint-terminal-Collect:
  assumes \( s \neq t \)
  assumes \( \forall \, x \, x'. \, S \, x \land T \, x' \implies x = x' \implies s = t \)
  shows \( \{ x. \, S \, x \} \cap \{ x. \, T \, x \} = \{ \} \)
using \, \text{assms by} \, \text{auto}

lemma disjoint-terminal:
  \( s \neq t \implies (\forall \, x'. \, x \in S \land x' \in T \implies x = x' \implies s = t) \implies S \cap T = \{ \} \)
by \, \text{blast}

lemma elim-singleton:
  assumes \( x \in \{ s \} \land x' \in \{ t \} \)
  obtains \( x = s \land x' = t \)
using \, \text{assms by} \, \text{blast}

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

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-ge-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\} ≥ (λP. \{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 partition-on (insert a A) P; have a ∈ (THE X. a ∈ X ∧ X ∈ P)
        using partition-on-in-the-unique-part by fastforce
      from partition-on (insert a A) P; have (THE X. a ∈ X ∧ X ∈ P) /∈ P
using partition-on-the-part-mem by fastforce
from this (partition-on (insert a A) P) have \((\text{T. } a \in X \land X \in P) - \{a\} \notin P\)
using partition-subset-imp-notin \(\langle a \in (\text{T. } a \in X \land X \in P) \rangle \) by blast
have \((\text{T. } a \in X \land X \in P) \neq \{a\}\)
using \(\langle \text{T. } a \in X \land X \in P \rangle \in P, \{a\} \notin P \rangle \) by auto
from partition-on (insert a A) P have \((\text{T. } a \in X \land X \in P) \subseteq insert a A\)
using \(\langle \text{T. } a \in X \land X \in P \rangle \in P\), partition-onD1 by fastforce
note facts-on-the-part-of = \(\langle a \in (\text{T. } a \in X \land X \in P) \rangle \) (\(\text{T. } a \in X \land X \in P) - \{a\} \notin P\)
\((\text{T. } a \in X \land X \in P) \subseteq insert a A\)
from partition-on (insert a A) P \(\langle \text{finite } A \rangle \) have \(\text{finite } P\)
by (meson finite.insert1 finite-elements)
from partition-on (insert a A) P \(\langle \text{finite } A \rangle \) have partition-on \((A - \{p\}) (P - \{\{p\}\})\)
using facts-on-the-part-of by (auto intro: partition-on-remove-singleton)
from this have partition-on A \?P'
using facts-on-the-part-of by (auto intro: partition-on-insert simp add: disjoint-iff)
moreover have \(\text{card } \{\text{?P'}\} = \text{Suc } k\)
proof
  from \(\text{card } P = \text{Suc } k\) have \(\text{card } (P - \{\text{T. } a \in X \land X \in P\}) = k\)
  using \(\text{finite } P\) \(\langle \text{T. } a \in X \land X \in P \rangle \in P\) by simp
  from this show \(\text{thesis}\)
  using \(\text{finite } P\) \(\langle \text{T. } a \in X \land X \in P \rangle - \{a\} \notin P\) by (simp add: card-insert-iff)
  qed
moreover have \(\text{?p} \in \{\text{?P'}\}\) by auto
moreover have \(P = \text{insert } (\text{insert } a \text{?p}) (\text{?P' - } \{\text{?p}\})\)
using facts-on-the-part-of by (auto simp add: insert-absorb)
ultimately have \(P \in \{P. \text{partition-on } A \text{?P} \land \text{card } P = \text{Suc } k\} \supseteq (\lambda P. P)\)
by auto
  then show \(\text{thesis}\) by auto
  qed
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 \(P = \text{insert } (\text{insert } a \text{?p}) (\text{?P' - } \{\text{?p}\})\)
  and partition-on A \text{?P'} and \(\text{card } P' = \text{Suc } k\) and \(p \in P'\) by auto
  from \(\langle p \in P'\rangle \) partition-on A \text{?P'} have partition-on \((A - p) (P' - \{p\})\)
end
proof
Suc k
injectivity-subexpr1
lemma
qed
by (auto)
card P
from assms
this
from assms
from assms
assumes
insert
assumes
assumes
assumes
assumes
ultimately show P ∈ ?S by auto
next
assume P ∈ ?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') (finite A) have finite P
using (P = insert {a} P') finite-elements by auto
from (partition-on A P') (a \notin A) have {a} \notin P'
using partition-onD1 by fastforce
from (P = insert {a} P') (card P' = k) this (finite P) have card P = Suc k
by auto
moreover from (partition-on A P') (a \notin A) have partition-on (insert a A) P
using (P = insert {a} P') by (simp add: partition-on-insert-singleton)
ultimately show P ∈ ?S by auto
qed

lemmas
inj
assumes a \notin A
assumes X \in P \land X' \in P'
assumes insert (insert a X) (P - {X}) = insert (insert a X') (P' - {X'})
assumes (partition-on A P \land card P = Suc k') \land (partition-on A P' \land card P'
= Suc k')
shows P = P' \land 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 insertI1 mem-simps(9) partition-onD1)
from this [a \notin X', a \notin X] show X = X'
by (meson insert-ident)
from assms(2, 3) show P = P'
using (insert a X = insert a X') \land (insert a X \notin P) \land (insert a X' \notin P')
by (metis insert-Diff insert-absorb insert-commute insert-ident)
qed
lemma injectivity-subexpr2:
assumes a /\notin A
assumes insert \{ a \} P = insert \{ a \} P'
assumes \( \text{partition-on} \ A \ P \land \text{card} \ P = k \) \land \( \text{partition-on} \ A \ P' \land \text{card} \ P' = k' \)
shows P = P'

proof –
  from assms(1, 3) have \{ a \} /\notin P \and \{ a \} /\notin P'
  using partition-onD1 by auto
  from \{ a \} /\notin P \have P = insert \{ a \} P - \{ \{ a \} \} by simp
  also from \{ insert \{ a \} P = insert \{ a \} P' \} have \ldots = insert \{ a \} P' - \{ \{ a \} \} by simp
  also from \{ a \} /\notin P' \have \ldots = P' by simp
finally show \?thesis .
qed

theorem card-partition-on:
assumes finite A
shows \text{card} \{ P. \text{partition-on} \ A \ P \land \text{card} \ P = k \} = \text{Stirling} (\text{card} A) k

using assms

proof (induct A arbitrary: k)
case empty
  have eq: \{ P. P = \{ \} \land \text{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 \{ insert a \} P = \{ insert \{ a \} P - \{ \} \} by simp
    have \text{eq}: \{ P. \text{partition-on} \ insert \{ a \} \ (\text{partition-on} \ A \ P \land \text{card} \ P = k' \) \land \text{card} \ P' = k' \} = \{ \}
    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. \text{partition-on} \ A \ P \land \text{card} \ P = \text{Suc} k' \};
    p <- P;
    \{ insert (insert a p) \ (P - \{p\}) \}
  }
  let \?subexpr2 = do {
    P <- \{ P. \text{partition-on} \ A \ P \land \text{card} \ P = k' \};
    \{ insert \{ a \} P \}
  }
  let \?expr = ?subexpr1 \cup ?subexpr2
  have \text{card} \{ P. \text{partition-on} \ (\text{insert} \ a \ A) \ P \land \text{card} \ P = k \} = \text{card} ?expr
  using finite A \a /\notin A \land \text{Suc} k' \by (simp add: set-partition-on-insert-with-fixed-card-eq)
  also have \text{card} ?expr = \text{Stirling} (\text{card} A) k' + \text{Stirling} (\text{card} A) (\text{Suc} k') \ast \text{Suc} k'

proof –
  have finite ?subexpr1 ∧ card ?subexpr1 = Stirling (card A) (Suc k') * Suc k'
    proof –
      from (finite A) have finite {P. partition-on A P ∧ card P = Suc k'}
        by (simp add: finitely-many-partition-on)
      moreover have \( \forall X \in \{P. \text{partition-on} A P \land \text{card P} = \text{Suc k'}\}. \text{finite} (X \gg (\lambda p. \{\text{insert} (\text{insert a p}) (X - \{p\}))))) \)
        using finite-elements (finite A) finite-bind
        by (metis (no-types, lifting) finite.emptyI finite.insert mem_Collect_eq)
      moreover have disjoint-family-on (\( \lambda P. \text{P} \gg (\lambda p. \{\text{insert} (\text{insert a p}) (P - \{p\})\}))\)
        by (injectivity-solver rule: injectivity-subexpr1(1)[OF \( a \notin A\)])
      moreover have card (\( \text{P} \gg (\lambda p. \{\text{insert} (\text{insert a p}) (P - \{p\})\})\)) = Suc k'
        if \( P \in \{P. \text{partition-on} A P \land \text{card P} = \text{Suc k'}\}\) for \( P \)
        proof –
          from (finite A) have finite \( P \)
            using finite-elements by blast
          moreover have inj-on (\( \lambda p. \{\text{insert} (\text{insert a p}) (P - \{p\})\}) \( P \)
            using that injectivity-subexpr1(2)[OF \( a \notin A\)] by (simp add: inj-onI)
          moreover from that have card \( P = \text{Suc k'}\) by simp
          ultimately show ?thesis by (simp add: card-bind-singleton)
        qed
      ultimately have card ?subexpr1 = card \( \{P. \text{partition-on} A P \land \text{card P} = \text{Suc k'}\}\) * Suc k'
        by (subst card-bind-constant) simp+
      from this have card ?subexpr1 = Stirling (card A) (Suc k') * Suc k'
        using insert.hyps(3) by simp
      moreover have finite ?subexpr1
        using (finite \( \{P. \text{partition-on} A P \land \text{card P} = \text{Suc k'}\}\)
        \( \forall X \in \{P. \text{partition-on} A P \land \text{card P} = \text{Suc k'}\}. \text{finite} (X \gg (\lambda p. \{\text{insert} (\text{insert a p}) (X - \{p\}))))) \)
        by (auto intro: finite-bind)
        ultimately show ?thesis by blast
    qed
  moreover have finite ?subexpr2 ∧ card ?subexpr2 = Stirling (card A) k'
    proof –
      from (finite A) have finite \( \{P. \text{partition-on} A P \land \text{card P} = k'\}\)
        by (simp add: finitely-many-partition-on)
      moreover have inj-on (\( \{\text{insert} \{a\}\}) \{P. \text{partition-on} A P \land \text{card P} = k'\}\)
        using injunctivity-subexpr2[OF \( a \notin A\)] by (simp add: inj-on-def)
      ultimately have card ?subexpr2 = card \( \{P. \text{partition-on} A P \land \text{card P} = k'\}\)
        by (simp add: card-bind-singleton)
      also have \( \ldots = \text{Stirling (card A) k'}\)
        using insert.hyps(3) .
      finally have card ?subexpr2 = Stirling (card A) k'.
    moreover have finite ?subexpr2
      by (simp add: finite \( \{P. \text{partition-on} A P \land \text{card P} = k'\}\) finite-bind)
  qed
ultimately show \( \text{thesis by blast} \)

qed

moreover have \( \text{?subexpr1} \cap \text{?subexpr2} = \{\} \)

proof –
  have \( \forall P \in \text{?subexpr1} \cdot \{a\} \notin P \)
    using insert1.hyps(2) by (force elim!: partition-onE)
  moreover have \( \forall P \in \text{?subexpr2} \cdot \{a\} \in P \) by auto
  ultimately show \( \text{?subexpr1} \cap \text{?subexpr2} = \{\} \) by blast

qed

ultimately show \( \text{thesis} \)

by (simp add: card-Un-disjoint)

qed

also have \( \dots = \text{Stirling (card (insert a A))} k \)

using insert(1, 2) \( \langle k = \text{Suc k'} \rangle \) by simp

finally show \( \text{thesis} \).

qed

theorem card-partition-on-at-most-size:
  assumes finite A
  shows \( \text{card } \{P. \text{partition-on } A\ P \land \text{card } P \leq k\} = (\sum j \leq k. \text{Stirling (card } A\ j)\) 

proof –
  have \( \text{card } \{P. \text{partition-on } A\ P \land \text{card } P \leq k\} = \text{card } (\bigcup j \leq k. \{P. \text{partition-on } A\ P \land \text{card } P = j\}) \)
    by (rule arg-cong[where \( f=\text{card} \)] auto
  also have \( \ldots = (\sum j \leq k. \text{card } \{P. \text{partition-on } A\ P \land \text{card } P = j\}) \)
    by (subst card-UN-disjoint) (auto simp add: finite A; finitely-many-partition-on)
  also have \( (\sum j \leq k. \text{card } \{P. \text{partition-on } A\ P \land \text{card } P = j\}) = (\sum j \leq k. \text{Stirling (card } A\ j) \)
    using (finite A) by (simp add: card-partition-on)
  finally show \( \text{thesis} \).

qed

theorem partition-on-size1:
  assumes finite A
  shows \( \{P. \text{partition-on } A\ P \land (\forall X \in P. \text{card } X = 1)\} = \{\langle \text{\lambda a}. \{a\} \rangle \cdot A\} \)

proof
  fix \( P \)
  assume \( P: P \in \{P. \text{partition-on } A\ P \land (\forall X \in P. \text{card } X = 1)\} \)
  have \( P = (\langle \text{\lambda a}. \{a\} \rangle \cdot A) \)

proof
  show \( P \subseteq (\langle \text{\lambda a}. \{a\} \rangle \cdot A) \)

proof
  fix \( X \)
  assume \( X \in P \)
  from \( P \) this obtain \( x \) where \( X = \{x\} \)
    by (auto simp add: card-Suc-eq)
from this \((X \in P)\) have \(x \in A\)

using \(P\) unfolding partition-on-def by blast
from this \(\langle X = \{x\}\rangle\) show \(X \in (\lambda a. \{a\}) \cdot A\) 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 \(X = \{x\} \in A\) by auto

have \(\bigcup P = A\)

using \(P\) unfolding partition-on-def by blast
from this \(\langle x \in A\rangle\) obtain \(X'\) where \(x \in X'\) and \(X' \in P\)

using UnionE by blast
from this \(\langle X' \in P\rangle\) have \(\operatorname{card} X' = 1\)

using \(P\) unfolding partition-on-def by auto

from this \(\langle X' \in P\rangle\) show \(X' \in P\) by auto

qed

qed

from this show \(P \in \{(\lambda a. \{a\}) \cdot A\}\) by auto

qed

next

show \(\{(\lambda a. \{a\}) \cdot A\} \subseteq \{P.\ partition-on \ A \ P \land (\forall X \in P. \ \operatorname{card} X = 1)\}\)

proof

fix \(P\)

assume \(P \in \{(\lambda a. \{a\}) \cdot A\}\)

from this have \(P = (\lambda a. \{a\}) \cdot A\) by auto

from this have partition-on \(A \ P\) by (auto intro: partition-onI)

from \(P\) this show \(P \in \{P.\ partition-on \ A \ P \land (\forall X \in P. \ \operatorname{card} X = 1)\}\) by auto

qed

qed

theorem card-partition-on-size1:

assumes \(\text{finite} \ A\)

shows \(\operatorname{card} \{P.\ partition-on \ A \ P \land (\forall X \in P. \ \operatorname{card} X = 1)\} = 1\)

using assms partition-on-size1 by fastforce

lemma card-partition-on-size1-eq-1:

assumes \(\text{finite} \ A\)

assumes \(\operatorname{card} A \leq k\)

shows \(\operatorname{card} \{P.\ partition-on \ A \ P \land \operatorname{card} P \leq k \land (\forall X \in P. \ \operatorname{card} X = 1)\} = 1\)

proof –

\{ fix \(P\)

assume partition-on \(A \ P \forall X \in P. \ \operatorname{card} X = 1\)

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

lemma card-partition-on-size1-eq-0:
assumes \( \text{finite}\ A \)
assumes \( k < \text{card}\ A \)
shows \( \text{card}\ \{P.\ \text{partition-on}\ A\ P \land \text{card}\ P \leq k \land (\forall X \in P.\ \text{card}\ X = 1)\} = 0 \)
proof –
{  
  fix \( P \)
  assume \( \text{partition-on}\ A\ P \land (\forall X \in P.\ \text{card}\ X = 1) \)
  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 \( \text{thesis} \) by (simp only: card.empty)
qed

end

References

