

# Cardinality of Set Partitions

Lukas Bulwahn

December 7, 2022

## 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

<b>1</b>	<b>Set Partitions</b>	<b>1</b>
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
<b>2</b>	<b>Combinatorial Basics</b>	<b>12</b>
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
<b>3</b>	<b>Cardinality of Set Partitions</b>	<b>15</b>

## 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**  $\bigwedge x. x \in A \implies x \in B$   
**assumes**  $\bigwedge x. x \in B \implies x \in A$   
**shows**  $A = B$   
**using** *assms* **by** *auto*

**lemma** *comp-image*:  
 $((\cdot) f \circ (\cdot) g) = (\cdot) (f \circ g)$   
**by** *rule auto*

## 1.2 Introduction and Elimination Rules

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

**lemma** *partition-onI*:  
**assumes**  $\bigwedge p. p \in P \implies p \neq \{\}$   
**assumes**  $\bigcup P = A$   
**assumes**  $\bigwedge p p'. p \in P \implies p' \in P \implies p \neq p' \implies p \cap p' = \{\}$   
**shows** *partition-on*  $A P$   
**using** *assms* **unfolding** *partition-on-def disjoint-def* **by** *blast*

**lemma** *partition-onE*:  
**assumes** *partition-on*  $A P$   
**obtains**  $\bigwedge p. p \in P \implies p \neq \{\}$   
 $\bigcup P = A$   
 $\bigwedge p p'. p \in P \implies p' \in P \implies p \neq p' \implies p \cap p' = \{\}$   
**using** *assms* **unfolding** *partition-on-def disjoint-def* **by** *blast*

## 1.3 Basic Facts on Set Partitions

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

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

**moreover note**  $\langle X' \subset X \rangle$   
**ultimately show** *False*  
**by** (*meson all-not-in-conv disjnt-iff psubsetD*)  
**qed**

**lemma** *partition-on-Diff*:  
**assumes** *P: partition-on A P* **shows**  $Q \subseteq P \implies \text{partition-on } (A - \bigcup Q) (P - Q)$   
**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:  $\bigwedge b. b \in B \implies \text{partition-on } b (P b)$*   
**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**  $i j \langle p \neq q \rangle$  *B[THEN partition-onD2, of i]* **by** (*simp add: disjointD*)  
**next**  
**assume**  $i \neq j$   
**then have** *disjnt 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: disjnt-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*  $A Q$   
**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** *disjnt*  $A X X \neq \{\}$   
**assumes**  $A \cup X = A'$   
**shows** *partition-on*  $A'$  (*insert*  $X P$ )  
**using** *assms* **by** (*auto simp: partition-on-def disjoint-def disjnt-def*)

An alternative formulation of  $\llbracket \text{partition-on } ?A \ ?P; \text{disjnt } ?A \ ?X; ?X \neq \{\} \rrbracket$ ;  
 $?A \cup ?X = ?A \rrbracket \implies \text{partition-on } ?A' \ (\text{insert } ?X \ ?P)$

**lemma** *partition-on-insert'*:  
**assumes** *partition-on*  $(A - X) P$   
**assumes**  $X \subseteq A X \neq \{\}$   
**shows** *partition-on*  $A$  (*insert*  $X P$ )  
**proof** –  
**have** *disjnt*  $(A - X) X$  **by** (*simp add: disjnt-iff*)  
**from** *assms*(1) *this* *assms*(2) **have** *partition-on*  $((A - X) \cup X)$  (*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*  $A P a \notin A$  *insert*  $a A = A'$   
**shows** *partition-on*  $A'$  (*insert*  $\{a\} P$ )  
**using** *assms* **by** (*auto simp: partition-on-def disjoint-def disjnt-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**  $\exists! X. x \in X \wedge X \in P$   
**proof** –

```

from ⟨partition-on A P⟩ have  $\bigcup P = A$ 
  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  $\exists! X. x \in X \wedge X \in P$ 
    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) theI)
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  $\exists! X. x \in X \wedge X \in P$ 
    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 ∈ 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  $\exists! X. x \in X \wedge X \in P$ 
    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  $x \in A$ 
shows  $(THE X. x \in X \wedge X \in P) = \{y. \exists X \in P. x \in X \wedge y \in X\}$ 
proof
show  $(THE X. x \in X \wedge X \in P) \subseteq \{y. \exists X \in P. x \in X \wedge y \in X\}$ 
proof
  fix  $y$ 
  assume  $y \in (THE X. x \in X \wedge X \in P)$ 
  moreover from  $\langle x \in A \rangle$  have  $x \in (THE X. x \in X \wedge X \in P)$ 
    using  $\langle partition-on A P \rangle$  partition-on-in-the-unique-part by force
  moreover from  $\langle x \in A \rangle$  have  $(THE X. x \in X \wedge X \in P) \in P$ 
    using  $\langle partition-on A P \rangle$  partition-on-the-part-mem by force
  ultimately show  $y \in \{y. \exists X \in P. x \in X \wedge y \in X\}$  by auto
qed
next
show  $\{y. \exists X \in P. x \in X \wedge y \in X\} \subseteq (THE X. x \in X \wedge X \in P)$ 
proof
  fix  $y$ 
  assume  $y \in \{y. \exists X \in P. x \in X \wedge y \in X\}$ 
  from this obtain  $X$  where  $x \in X$  and  $y \in X$  and  $X \in P$  by auto
  from  $\langle x \in X \rangle \langle X \in P \rangle$  have  $(THE X. x \in X \wedge X \in P) = X$ 
    using  $\langle partition-on A P \rangle$  partition-on-the-part-eq by force
  from this  $\langle y \in X \rangle$  show  $y \in (THE X. x \in X \wedge X \in P)$  by simp
qed
qed

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

lemma partition-on-part-characteristic:
  assumes partition-on A P
  assumes  $X \in P$   $x \in X$ 

```

**shows**  $X = \{y. \exists X \in P. x \in X \wedge y \in X\}$   
**proof** –  
**from**  $\langle x \in X \rangle \langle X \in P \rangle$  **have**  $x \in A$   
**using**  $\langle \text{partition-on } A \ P \rangle \text{ partition-onE}$  **by** *blast*  
**from**  $\langle x \in X \rangle \langle X \in P \rangle$  **have**  $X = (\text{THE } X. x \in X \wedge X \in P)$   
**using**  $\langle \text{partition-on } A \ P \rangle$  **by** (*simp add: partition-on-the-part-eq*)  
**also from**  $\langle x \in A \rangle$  **have**  $(\text{THE } X. x \in X \wedge X \in P) = \{y. \exists X \in P. x \in X \wedge y \in X\}$   
**using**  $\langle \text{partition-on } A \ P \rangle$  *the-unique-part-alternative-def* **by** *force*  
**finally show** *?thesis* .  
**qed**

**lemma** *partition-on-no-partition-outside-carrier*:  
**assumes** *partition-on A P*  
**assumes**  $x \notin A$   
**shows**  $\{y. \exists X \in P. x \in X \wedge y \in X\} = \{\}$   
**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*:  $\forall p \in P. p \neq \{\} \cup P = \text{insert } a \ A$   
 $\forall p \in P. \forall p' \in P. p \neq p' \longrightarrow p \cap p' = \{\}$   
**using**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$  **by** (*fastforce elim: partition-onE*)  
**from** *this*(2, 3)  $\langle a \notin A \rangle \langle \{a\} \in P \rangle$  **have** *A-eq*:  $A = \bigcup (P - \{\{a\}\})$   
**by** *auto* (*metis Int-iff UnionI empty-iff insert-iff*)  
**from** *prop-partition-on A-eq* **have** *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)  $\langle \{a\} \in P \rangle$  **show** *?thesis*  
**using** *finite-elements[OF \{finite A\} partition-on]* **by** *simp*  
**next**  
**case** *False*  
**from**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$  **obtain** *p* **where** *p-def*:  $p \in P \ a \in p$   
**by** (*blast elim: partition-onE*)  
**from**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$  *p-def* **have** *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  $p - \{a\} \notin P$   
 unfolding partition-on-def disjoint-def  
 by (metis Diff-insert-absorb Diff-subset inf.orderE mk-disjoint-insert)  
 let  $?P' = \text{insert } (p - \{a\}) (P - \{p\})$   
 have partition-on A  $?P'$   
 proof (rule partition-onI)  
 from ⟨partition-on (insert a A) P⟩ have  $\forall p \in P. p \neq \{\}$  by (auto elim:  
 partition-onE)  
 from this p-def ⟨ $\{a\} \notin P$ ⟩ show  $\bigwedge p'. p' \in \text{insert } (p - \{a\}) (P - \{p\}) \implies p' \neq \{\}$   
 by (simp; metis (no-types) Diff-eq-empty-iff subset-singletonD)  
 next  
 from ⟨partition-on (insert a A) P⟩ have  $\bigcup P = \text{insert } a A$  by (auto elim:  
 partition-onE)  
 from p-def this ⟨ $a \notin A$ ⟩ a-notmem show  $\bigcup (\text{insert } (p - \{a\}) (P - \{p\})) = A$  by auto  
 next  
 show  $\bigwedge pa pa'. pa \in \text{insert } (p - \{a\}) (P - \{p\}) \implies pa' \in \text{insert } (p - \{a\}) (P - \{p\}) \implies pa \neq pa' \implies pa \cap pa' = \{\}$   
 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  $\dots = \text{card } ?P'$  using ⟨ $p - \{a\} \notin P$ ⟩ ⟨finite P⟩ by simp  
 also have  $\dots \leq \text{card } A$  using ⟨partition-on A  $?P'$ ⟩ insert.hyps(3) by simp  
 also have  $\dots \leq \text{card } (\text{insert } a A)$  by (simp add: card-insert-le ⟨finite A⟩)  
 finally show ?thesis .  
 qed  
 qed

## 1.6 Operations on Set Partitions

lemma partition-on-union:

assumes  $A \cap B = \{\}$   
 assumes partition-on A P  
 assumes partition-on B Q  
 shows partition-on (A ∪ B) (P ∪ Q)

proof (rule partition-onI)

fix X

assume  $X \in P \cup Q$

from this ⟨partition-on A P⟩ ⟨partition-on B Q⟩ show  $X \neq \{\}$

by (auto elim: partition-onE)

next

show  $\bigcup (P \cup Q) = A \cup B$

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  $\bigcup P = A \cup C$ 
  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  $f \in A \rightarrow_E P$ 
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  $\langle$ partition-on  $B P\rangle$  show  $X \neq \{\}$ 
    by (auto elim: partition-onE)
next
  show  $\bigcup ?P = A \cup B$ 
    using  $\langle$ partition-on  $B P\rangle$   $\langle$  $f \in A \rightarrow_E P\rangle$  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  $X': X = X' \cup \{x \in A. f x = X'\}$   $X' \in P$  by
auto
  from  $\langle$  $Y \in ?P\rangle$  obtain  $Y'$  where  $Y': Y = Y' \cup \{x \in A. f x = Y'\}$   $Y' \in P$ 
by auto
  from  $\langle$  $X \neq Y\rangle$   $X' Y'$  have  $X' \neq Y'$  by auto
  from this  $X' Y'$  have  $X' \cap Y' = \{\}$ 
    using  $\langle$ partition-on  $B P\rangle$  by (auto elim!: partition-onE)
  from  $X' Y'$  have  $X' \subseteq B Y' \subseteq B$ 
    using  $\langle$ partition-on  $B P\rangle$  by (auto elim!: partition-onE)
  from this  $\langle$  $X' \cap Y' = \{\}\rangle$   $X' Y'$   $\langle$  $X' \neq Y'\rangle$  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  $(f ' A) ((\cdot) f ' P)$ 
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  $(f ' A) ((\cdot) f ' P)$ 
    by (auto intro!: partition-onI elim!: partition-onE)
qed

```

**lemma** *set-of-partition-on-map*:

```

assumes inj-on  $f A$ 

```

```

shows (⋅) ((⋅) f) ‘ {P. partition-on A P} = {P. partition-on (f ‘ A) P}
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: ⋀X x. X ∈ P ⇒ x ∈ X ⇒ x ∈ f ‘ A
  by (auto elim!: partition-onE)
have (⋅) (f ∘ the-inv-into A f) ‘ P = (⋅) f ‘ (⋅) (the-inv-into A f) ‘ P
  by (simp add: image-image cong: image-cong-simp)
moreover have P = (⋅) (f ∘ the-inv-into A f) ‘ P
proof (rule set-eqI′)
  fix X
  assume X: X ∈ P
  moreover from X mem have in-range: ∀x∈X. x ∈ f ‘ A by auto
  moreover have X = (f ∘ the-inv-into A f) ‘ X
  proof (rule set-eqI′)
    fix x
    assume x ∈ X
    show x ∈ (f ∘ the-inv-into A f) ‘ X
    proof (rule image-eqI)
      from in-range ⟨x ∈ X⟩ assms show x = (f ∘ 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 ∘ 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 ∘ the-inv-into A f) ‘ P by auto
next
fix X
assume X ∈ (⋅) (f ∘ the-inv-into A f) ‘ P
moreover
{
  fix Y
  assume Y ∈ P
  from this ⟨inj-on f A⟩ mem have ∀x∈Y. f (the-inv-into A f x) = x
    by (auto simp add: f-the-inv-into-f)
  from this have (f ∘ the-inv-into A f) ‘ Y = Y by force
}

```

```

    ultimately show  $X \in P$  by auto
qed
ultimately have  $P: P = (\cdot) f \text{ ' } (\cdot) (\text{the-inv-into } A f) \text{ ' } P$  by simp
have  $A\text{-eq}: A = \text{the-inv-into } A f \text{ ' } f \text{ ' } A$  by (simp add: assms)
from  $\langle \text{inj-on } f A \rangle$  have  $\text{inj-on } (\text{the-inv-into } A f) (f \text{ ' } A)$ 
  using  $\langle \text{partition-on } (f \text{ ' } A) P \rangle$  by (simp add: inj-on-the-inv-into)
from this have  $(\cdot) (\text{the-inv-into } A f) \text{ ' } P \in \{P. \text{partition-on } A P\}$ 
  using  $\langle \text{partition-on } (f \text{ ' } A) P \rangle$  by (subst A-eq, auto intro!: partition-on-map)
from  $P$  this show  $P \in (\cdot) ((\cdot) f) \text{ ' } \{P. \text{partition-on } A P\}$  by (rule image-eqI)
qed

end

```

## 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  $\text{inj-on } f A$ 
  shows  $\text{disjoint-family-on } (\lambda x. \{f x\}) A$ 
using assms disjoint-family-on-def inj-on-contraD by fastforce

```

#### 2.1.2 Cardinality Theorems for Set.bind

```

lemma card-bind:
  assumes  $\text{finite } S$ 
  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  $\text{card } (S \gg= f) = \text{card } (\bigcup (f \text{ ' } S))$ 
    by (simp add: bind-UNION)
  also have  $\text{card } (\bigcup (f \text{ ' } S)) = (\sum x \in S. \text{card } (f x))$ 
    using assms unfolding disjoint-family-on-def by (simp add: card-UN-disjoint)
  finally show ?thesis .
qed

```

**lemma** *card-bind-constant*:  
**assumes** *finite S*  
**assumes**  $\forall X \in S. \text{finite } (f X)$   
**assumes** *disjoint-family-on f S*  
**assumes**  $\bigwedge x. x \in S \implies \text{card } (f x) = k$   
**shows**  $\text{card } (S \ggg f) = \text{card } S * k$   
**using** *assms* **by** (*simp add: card-bind*)

**lemma** *card-bind-singleton*:  
**assumes** *finite S*  
**assumes** *inj-on f S*  
**shows**  $\text{card } (S \ggg (\lambda x. \{f x\})) = \text{card } S$   
**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**  $\bigwedge i j. i \in I \wedge j \in I \implies i \neq j \implies (A i) \cap (A j) = \{\}$   
**shows** *disjoint-family-on A I*  
**using** *assms* **unfolding** *disjoint-family-on-def* **by** *auto*

**lemma** *disjoint-bind*:  $\bigwedge S T f g. (\bigwedge s t. S s \wedge T t \implies f s \cap g t = \{\}) \implies (\{s. S s\} \ggg f) \cap (\{t. T t\} \ggg g) = \{\}$   
**by** *fastforce*

**lemma** *disjoint-bind'*:  $\bigwedge S T f g. (\bigwedge s t. s \in S \wedge t \in T \implies f s \cap g t = \{\}) \implies (S \ggg f) \cap (T \ggg g) = \{\}$   
**by** *fastforce*

```

lemma injectivity-solver-CollectE:
  assumes  $a \in \{x. P\ x\} \wedge a' \in \{x. P'\ x\}$ 
  assumes  $(P\ a \wedge P'\ a') \implies W$ 
  shows  $W$ 
using assms by auto

lemma injectivity-solver-prep-assms-Collect:
  assumes  $x \in \{x. P\ x\}$ 
  shows  $P\ x \wedge P'\ x$ 
using assms by simp

lemma injectivity-solver-prep-assms:  $x \in A \implies x \in A \wedge x \in A$ 
  by simp

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

lemma disjoint-terminal-Collect:
  assumes  $s \neq t$ 
  assumes  $\bigwedge x\ x'. S\ x \wedge T\ x' \implies x = x' \implies s = t$ 
  shows  $\{x. S\ x\} \cap \{x. T\ x\} = \{\}$ 
using assms by auto

lemma disjoint-terminal:
   $s \neq t \implies (\bigwedge x\ x'. x \in S \wedge x' \in T \implies x = x' \implies s = t) \implies S \cap T = \{\}$ 
by blast

lemma elim-singleton:
  assumes  $x \in \{s\} \wedge x' \in \{t\}$ 
  obtains  $x = s \wedge x' = t$ 
using assms by blast

method injectivity-solver uses rule =
  insert method-facts,
  use nothing in <
  ((drule injectivity-solver-prep-assms-Collect | drule injectivity-solver-prep-assms) $+$ ) $?$ ;
  rule disjoint-family-onI;
  ((rule disjoint-bind | rule disjoint-bind') $+$ ) $?$ ;
  (erule elim-singleton) $?$ ;
  (erule disjoint-terminal-singleton | erule disjoint-terminal-Collect | erule disjoint-terminal);
  (elim injectivity-solver-CollectE) $?$ ;
  rule rule;
  assumption+
  >

end

```

### 3 Cardinality of Set Partitions

**theory** *Card-Partitions*

**imports**

*HOL-Combinatorics.Stirling*

*Set-Partition*

*Injectivity-Solver*

**begin**

**lemma** *set-partition-on-insert-with-fixed-card-eq*:

**assumes** *finite A*

**assumes**  $a \notin A$

**shows**  $\{P. \text{partition-on } (insert\ a\ A)\ P \wedge card\ P = Suc\ k\} = (do\ \{$

$P <- \{P. \text{partition-on } A\ P \wedge card\ P = Suc\ k\};$

$p <- P;$

$\{insert\ (insert\ a\ p)\ (P - \{p\})\}$

$\})$

$\cup\ (do\ \{$

$P <- \{P. \text{partition-on } A\ P \wedge card\ P = k\};$

$\{insert\ \{a\}\ P\}$

$\})$  **(is**  $?S = ?T$ )

**proof**

**show**  $?S \subseteq ?T$

**proof**

**fix**  $P$

**assume**  $P \in \{P. \text{partition-on } (insert\ a\ A)\ P \wedge card\ P = Suc\ k\}$

**from this have** *partition-on*  $(insert\ a\ A)\ P$  **and**  $card\ P = Suc\ k$  **by** *auto*

**show**  $P \in ?T$

**proof** *cases*

**assume**  $\{a\} \in P$

**have** *partition-on*  $A\ (P - \{\{a\}\})$

**using**  $\langle \{a\} \in P \rangle \langle \text{partition-on } (insert\ a\ A)\ P \rangle$  [*THEN partition-on-Diff, of*  $\{\{a\}\} \langle a \notin A \rangle$

**by** *auto*

**moreover from**  $\langle \{a\} \in P \rangle \langle card\ P = Suc\ k \rangle$  **have**  $card\ (P - \{\{a\}\}) = k$

**by** (*subst card-Diff-singleton*) (*auto intro: card-ge-0-finite*)

**moreover from**  $\langle \{a\} \in P \rangle$  **have**  $P = insert\ \{a\}\ (P - \{\{a\}\})$  **by** *auto*

**ultimately have**  $P \in \{P. \text{partition-on } A\ P \wedge card\ P = k\} \gg (\lambda P. \{insert\ \{a\}\ P\})$

**by** *auto*

**from this show** *?thesis* **by** *auto*

**next**

**assume**  $\{a\} \notin P$

**let**  $?p' = (THE\ X. a \in X \wedge X \in P)$

**let**  $?p = (THE\ X. a \in X \wedge X \in P) - \{a\}$

**let**  $?P' = insert\ ?p\ (P - \{?p'\})$

**from**  $\langle \text{partition-on } (insert\ a\ A)\ P \rangle$  **have**  $a \in (THE\ X. a \in X \wedge X \in P)$

**using** *partition-on-in-the-unique-part* **by** *fastforce*

**from**  $\langle \text{partition-on } (insert\ a\ A)\ P \rangle$  **have**  $(THE\ X. a \in X \wedge X \in P) \in P$

**using** *partition-on-the-part-mem* **by** *fastforce*  
**from** *this*  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$  **have**  $(\text{THE } X. a \in X \wedge X \in P) - \{a\} \notin P$   
**using** *partition-subset-imp-notin*  $\langle a \in (\text{THE } X. a \in X \wedge X \in P) \rangle$  **by** *blast*  
**have**  $(\text{THE } X. a \in X \wedge X \in P) \neq \{a\}$   
**using**  $\langle (\text{THE } X. a \in X \wedge X \in P) \in P \rangle$   $\langle \{a\} \notin P \rangle$  **by** *auto*  
**from**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$  **have**  $(\text{THE } X. a \in X \wedge X \in P) \subseteq \text{insert } a \ A$   
**using**  $\langle (\text{THE } X. a \in X \wedge X \in P) \in P \rangle$  *partition-onD1* **by** *fastforce*  
**note** *facts-on-the-part-of* =  $\langle a \in (\text{THE } X. a \in X \wedge X \in P) \rangle$   $\langle (\text{THE } X. a \in X \wedge X \in P) \in P \rangle$   
 $\langle (\text{THE } X. a \in X \wedge X \in P) - \{a\} \notin P \rangle$   $\langle (\text{THE } X. a \in X \wedge X \in P) \neq \{a\} \rangle$   
 $\langle (\text{THE } X. a \in X \wedge X \in P) \subseteq \text{insert } a \ A \rangle$   
**from**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$   $\langle \text{finite } A \rangle$  **have** *finite P*  
**by** *(meson finite.insertI finite-elements)*  
**from**  $\langle \text{partition-on } (\text{insert } a \ A) \ P \rangle$   $\langle a \notin A \rangle$  **have** *partition-on (A - ?p) (P - {?p}^ $\wedge$ )*  
**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: disjnt-iff)*  
**moreover** **have** *card ?P' = Suc k*  
**proof** -  
**from**  $\langle \text{card } P = \text{Suc } k \rangle$  **have** *card (P - {THE X. a \in X \wedge X \in P}) = k*  
**using**  $\langle \text{finite } P \rangle$   $\langle (\text{THE } X. a \in X \wedge X \in P) \in P \rangle$  **by** *simp*  
**from** *this* **show** *?thesis*  
**using**  $\langle \text{finite } P \rangle$   $\langle (\text{THE } X. a \in X \wedge X \in P) - \{a\} \notin P \rangle$  **by** *(simp add: card-insert-if)*  
**qed**  
**moreover** **have**  $?p \in ?P'$  **by** *auto*  
**moreover** **have**  $P = \text{insert } (\text{insert } a \ ?p) (?P' - \{?p\})$   
**using** *facts-on-the-part-of* **by** *(auto simp add: insert-absorb)*  
**ultimately** **have**  $P \in \{\lambda P. \text{partition-on } A \ P \wedge \text{card } P = \text{Suc } k\} \gg (\lambda P. P \gg (\lambda p. \{\text{insert } (\text{insert } a \ p) (P - \{p\})\}))$   
**by** *auto*  
**then** **show** *?thesis* **by** *auto*  
**qed**  
**qed**  
**next**  
**show**  $?T \subseteq ?S$   
**proof**  
**fix**  $P$   
**assume**  $P \in ?T$  **(is**  $- \in ?\text{subexpr1} \cup ?\text{subexpr2}$ **)**  
**from** *this* **show**  $P \in ?S$   
**proof**  
**assume**  $P \in ?\text{subexpr1}$   
**from** *this* **obtain**  $p \ P'$  **where**  $P = \text{insert } (\text{insert } a \ p) (P' - \{p\})$   
**and** *partition-on A P'* **and**  $\text{card } P' = \text{Suc } k$  **and**  $p \in P'$  **by** *auto*  
**from**  $\langle p \in P' \rangle$   $\langle \text{partition-on } A \ P' \rangle$  **have** *partition-on (A - p) (P' - {p})*



```

    by (simp add: partition-on-remove-singleton)
  from ⟨partition-on A P'⟩ ⟨finite A⟩ have finite P
    using ⟨P = -⟩ finite-elements by auto
  from ⟨partition-on A P'⟩ ⟨a ∉ A⟩ have insert a p ∉ P' - {p}
    using partition-onD1 by fastforce
  from ⟨P = -⟩ this ⟨card P' = Suc k⟩ ⟨finite P⟩ ⟨p ∈ P'⟩
  have card P = Suc k by auto
  moreover have partition-on (insert a A) P
    using ⟨partition-on (A - p) (P' - {p})⟩ ⟨a ∉ A⟩ ⟨p ∈ P'⟩ ⟨partition-on A
P'⟩ ⟨P = -⟩
    by (auto intro!: partition-on-insert dest: partition-onD1 simp add: disjnt-iff)
  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 ∉ A⟩ have {a} ∉ 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 ∉ 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
qed
qed

```

**lemma** *injectivity-subexpr1*:

```

  assumes a ∉ A
  assumes X ∈ P ∧ X' ∈ P'
  assumes insert (insert a X) (P - {X}) = insert (insert a X') (P' - {X'})
  assumes (partition-on A P ∧ card P = Suc k) ∧ (partition-on A P' ∧ card P'
= Suc k')
  shows P = P' and X = X'
proof -
  from assms(1, 2, 4) have a ∉ X a ∉ X'
    using partition-onD1 by auto
  from assms(1, 4) have insert a X ∉ P insert a X' ∉ 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 ∉ X'⟩ ⟨a ∉ X⟩ show X = X'
    by (meson insert-ident)
  from assms(2, 3) show P = P'
    using ⟨insert a X = insert a X'⟩ ⟨insert a X ∉ P⟩ ⟨insert a X' ∉ P'⟩
    by (metis insert-Diff insert-absorb insert-commute insert-ident)

```

qed

**lemma** *injectivity-subexpr2*:

assumes  $a \notin A$

assumes  $\text{insert } \{a\} P = \text{insert } \{a\} P'$

assumes  $(\text{partition-on } A P \wedge \text{card } P = k') \wedge \text{partition-on } A P' \wedge \text{card } P' = k'$

shows  $P = P'$

**proof** –

from *assms*(1, 3) have  $\{a\} \notin P$  and  $\{a\} \notin P'$

using *partition-onD1* by *auto*

from  $\langle \{a\} \notin P \rangle$  have  $P = \text{insert } \{a\} P - \{\{a\}\}$  by *simp*

also from  $\langle \text{insert } \{a\} P = \text{insert } \{a\} P' \rangle$  have  $\dots = \text{insert } \{a\} P' - \{\{a\}\}$  by *simp*

also from  $\langle \{a\} \notin P' \rangle$  have  $\dots = P'$  by *simp*

finally show *?thesis* .

qed

**theorem** *card-partition-on*:

assumes *finite A*

shows  $\text{card } \{P. \text{partition-on } A P \wedge \text{card } P = k\} = \text{Stirling } (\text{card } A) k$

using *assms*

**proof** (*induct A arbitrary: k*)

case *empty*

have *eq*:  $\{P. P = \{\} \wedge \text{card } P = 0\} = \{\{\}\}$  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*(1) have *empty*:  $\{P. \text{partition-on } (\text{insert } a A) P \wedge \text{card } P = 0\} = \{\}$

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 {

$P <- \{P. \text{partition-on } A P \wedge \text{card } P = \text{Suc } k'\};$

$p <- P;$

$\{\text{insert } (\text{insert } a p) (P - \{p\})\}$

}

let *?subexpr2* = do {

$P <- \{P. \text{partition-on } A P \wedge \text{card } P = k'\};$

$\{\text{insert } \{a\} P\}$

}

let *?expr* = *?subexpr1*  $\cup$  *?subexpr2*

have  $\text{card } \{P. \text{partition-on } (\text{insert } a A) P \wedge \text{card } P = k\} = \text{card } ?\text{expr}$

using  $\langle \text{finite } A \rangle \langle a \notin A \rangle \langle k = \text{Suc } k' \rangle$  by (*simp add: set-partition-on-insert-with-fixed-card-eq*)

also have  $\text{card } ?\text{expr} = \text{Stirling } (\text{card } A) k' + \text{Stirling } (\text{card } A) (\text{Suc } k') * \text{Suc}$

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

by (simp add: ⟨finite {P. partition-on A P ∧ card P = k'}⟩ finite-bind)  
 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  
 qed

**theorem 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

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

```

    by (auto simp add: card-Suc-eq)
  from this ⟨X ∈ P⟩ have x ∈ A
    using P unfolding partition-on-def by blast
  from this ⟨X = {x}⟩ show X ∈ (λa. {a}) ‘ A by auto
qed
next
show (λa. {a}) ‘ A ⊆ P
proof
  fix X
  assume X ∈ (λa. {a}) ‘ A
  from this obtain x where X: X = {x} x ∈ A by auto
  have ⋃ P = A
    using P unfolding partition-on-def by blast
  from this ⟨x ∈ A⟩ obtain X' where x ∈ X' and X' ∈ P
    using UnionE by blast
  from ⟨X' ∈ P⟩ have card X' = 1
    using P unfolding partition-on-def by auto
  from this ⟨x ∈ X'⟩ have X' = {x}
    using card-1-singletonE by blast
  from this X(1) ⟨X' ∈ P⟩ show X ∈ P by auto
qed
qed
from this show P ∈ {(λa. {a}) ‘ A} by auto
qed
next
show {(λa. {a}) ‘ A} ⊆ {P. partition-on A P ∧ (∀ X ∈ P. card X = 1)}
proof
  fix P
  assume P ∈ {(λa. {a}) ‘ A}
  from this have P: P = (λa. {a}) ‘ A by auto
  from this have partition-on A P by (auto intro: partition-onI)
  from P this show P ∈ {P. partition-on A P ∧ (∀ X ∈ P. card X = 1)} by auto
qed
qed

theorem card-partition-on-size1:
  assumes finite A
  shows card {P. partition-on A P ∧ (∀ X ∈ P. card X = 1)} = 1
using assms partition-on-size1 by fastforce

lemma card-partition-on-size1-eq-1:
  assumes finite A
  assumes card A ≤ k
  shows card {P. partition-on A P ∧ card P ≤ k ∧ (∀ X ∈ P. card X = 1)} = 1
proof -
  {
    fix P
    assume partition-on A P ∧ (∀ X ∈ P. card X = 1)
    from this have P ∈ {P. partition-on A P ∧ (∀ X ∈ P. card X = 1)} by simp
  }

```

```

from this have  $P \in \{(\lambda a. \{a\}) \text{ ` } A\}$ 
  using partition-on-size1  $\langle \text{finite } A \rangle$  by auto
from this have  $P = (\lambda a. \{a\}) \text{ ` } A$  by auto
moreover from this have  $\text{card } P = \text{card } A$ 
  by (auto intro: card-image)
}
from this have  $\{P. \text{partition-on } A \ P \wedge \text{card } P \leq k \wedge (\forall X \in P. \text{card } X = 1)\} =$ 
 $\{P. \text{partition-on } A \ P \wedge (\forall X \in P. \text{card } X = 1)\}$ 
  using  $\langle \text{card } A \leq k \rangle$  by auto
from this show ?thesis
  using  $\langle \text{finite } A \rangle$  by (simp only: card-partition-on-size1)
qed

```

**lemma** *card-partition-on-size1-eq-0*:

```

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

```

**end**

## References

- [1] F. Haftmann. Discrete summation. *Archive of Formal Proofs*, Apr. 2014. [http://isa-afp.org/entries/Discrete\\_Summation.shtml](http://isa-afp.org/entries/Discrete_Summation.shtml), Formal proof development.
- [2] R. P. Stanley. *Enumerative Combinatorics: Volume 1*. Cambridge University Press, second edition, 2012.
- [3] Wikipedia. Stirling numbers of the second kind — wikipedia, the free encyclopedia, 2015. <https://en.wikipedia.org/w/index.php?>

[title=Stirling\\_numbers\\_of\\_the\\_second\\_kind&oldid=693800357](#), [Online; accessed 12-December-2015].