Spivey’s Generalized Recurrence for Bell Numbers

Lukas Bulwahn

October 10, 2017

Abstract

This entry defines the Bell numbers [1] as the cardinality of set partitions for a carrier set of given size, and derives Spivey’s generalized recurrence relation for Bell numbers [2] following his elegant and intuitive combinatorial proof.

As the set construction for the combinatorial proof requires construction of three intermediate structures, the main difficulty of the formalization is handling the overall combinatorial argument in a structured way. The introduced proof structure allows us to compose the combinatorial argument from its subparts, and supports to keep track how the detailed proof steps are related to the overall argument. To obtain this structure, this entry uses set monad notation for the set construction’s definition, introduces suitable predicates and rules, and follows a repeating structure in its Isar proof.

Contents

1 Set Partitions 2
  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 . 3
  1.5 Cardinality of Parts in a Set Partition . . . . . . . . . . . . 5
  1.6 Operations on Set Partitions . . . . . . . . . . . . . . . . . 6

2 Bell Numbers and Spivey’s Generalized Recurrence 10
  2.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  2.1.1 Additions to FuncSet . . . . . . . . . . . . . . . . . . . 10
  2.1.2 Additions for Injectivity Proofs . . . . . . . . . . . . . 10
  2.1.3 Disjointness under Function Application . . . . . . . 11
  2.1.4 Cardinality Theorems for Set.bind . . . . . . . . . . . . 12
  2.2 Definition of Bell Numbers . . . . . . . . . . . . . . . . . . 13
  2.3 Construction of the Partitions . . . . . . . . . . . . . . . . 13
  2.4 Injectivity of the Set Construction . . . . . . . . . . . . . . 18
1 Set Partitions

theory Set-Partition

imports
  HOL-Library.FuncSet
  Card-Partitions.Card-Partitions

begin

1.1 Useful Additions to Main Theories

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

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

1.2 Introduction and Elimination Rules

The definition of partition-on is in Disjoint-Sets.

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

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

1.3 Basic Facts on Set Partitions

lemma partition-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 $A \cap B = \emptyset$
  shows $P \cap Q = \emptyset$
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)

An alternative formulation of $\left[ \text{partition-on } ?A ?B; \text{disjnt } ?A ?A'; ?A' \neq \{\} \right] \implies \text{partition-on } (?A \cup ?A') (\text{insert } ?A' ?B)$

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

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 $\langle \text{partition-on } A P \rangle$ have $\bigcup P = A$
  by (auto elim: partition-onE)
  from this $\langle x \in A \rangle$ obtain $X$ where $X: x \in X \land X \in P$ by blast
  |
  | fix $Y$
  | assume $x \in X \land Y \in P$
  | from this have $X = Y$
  | using $\langle \text{partition-on } A P \rangle$ 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 $(\text{THE } x. x \in X \land X \in P) \in P$
proof
from (\langle x \in A \rangle) have \exists!X. x \in X \land X \in P
using (\text{partition-on } A P) by (\text{simp add: partition-on-partition-on-unique})
from this show $(\text{THE } x. x \in X \land X \in P) \in P$
by (metis (no-types, lifting) theI)
qed

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

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

1.5 Cardinality of Parts in a Set Partition

lemma partition-on-le-set-elements:
assumes finite A
assumes partition-on A P
shows card P \( \leq \) card A
using assms
proof (induct A arbitrary: P)
case empty
from this show card P \( \leq \) 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:
\( \forall p \in P. \forall p' \in P. p \neq p' \rightarrow p \cap p' = \{\} \)
using partition-on (insert a A) P by (fastforce elim: partition-onE)+
from this(2, 3) \( a \notin A \); \{a\} \( \in \) P; 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 A \((P - \{\{a\}\})\)
by (intro partition-onI) auto
from insert.hyps(3) this have card \((P - \{\{a\}\})\) \( \leq \) 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: \( p \in P \); a \( \notin \) p'
by (blast elim: partition-onE)
from partition-on (insert a A) P: 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' = 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 \( \forall p'. p' \in \text{insert} \ (p - \{a\}) \ (P - \{p\}) \rightarrow p' \neq \{\} \)
by (simp; metis (no-types) Diff-eq-empty-iff subset-singletonD)
nextrom partition-on (insert a A) P: have \( \bigcup P = \text{insert} \ a \ A \)
by (auto elim: partition-onE)
1.6 Operations on Set Partitions

**Lemma partition-on-union:**
assumes \( A \cap B = \{\} \)
assembles \( \textit{partition-on} \ A \, P \)
assembles \( \textit{partition-on} \ B \, Q \)
shows \( \textit{partition-on} \ (A \cup B) \ (P \cup Q) \)
proof (rule partition-onI)
fix \( X \)
assume \( X \in P \cup Q \)
from this \( \textit{partition-on} \ A \, P \) \( \textit{partition-on} \ B \, Q \) show \( X \neq \{\} \)
by (auto elim: partition-onE)

text
fix \( X \,
assume \( X \in P \cup Q \)
\textit{Y} \in P \cup Q \, X \neq Y \)
from this asms show \( X \cap Y = \{\} \)
by (elim UnE partition-onE) auto

**Lemma partition-on-split1:**
assumes \( \textit{partition-on} \ A \ (P \cup Q) \)
shows \( \textit{partition-on} \ (\bigcup P) \ P \)
proof (rule partition-onI)
fix \( p \)
assume \( p \in P \)
from this asms 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 $C$ (($\lambda X. X \cap C) \ 
  \ 
  \cdot \ P$)
proof (rule partition-onI)
  fix $p$
  assume $p \in (\lambda X. X \cap C) \ 
  \cdot \ P$
  from this assms show $p \neq \{\}$ by auto
next
  have $\bigcup P = A \cup C$
  using $\langle$ partition-on $B$ $P$ $\rangle$
  from this show $\bigcup((\lambda X. X \cap C) \ 
  \cdot \ P) = C$ by auto
next
  fix $p \ p'$
  assume $p \in (\lambda X. X \cap C) \ 
  \cdot \ P \ p' \in (\lambda X. X \cap C) \ 
  \cdot \ P \ p \neq p'$
  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 ($A \cup B$) (($\lambda X. X \cup \{x \in A. f x = X\}) \ 
  \cdot \ P$) (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 $B$ $P); (f \in A \rightarrow_E P); by (auto elim: partition-onE)$
next
fix X Y
assume X ∈ ?P Y ∈ ?P X ≠ Y
from ⟨X ∈ ?P⟩ obtain X′ where X′: X = X′ ∪ {x ∈ A. f x = X′} X′ ∈ P by auto
doi: ⟨X ∈ ?P⟩ obtain X′ where X′: Y = Y′ ∪ {x ∈ A. f x = Y′} Y′ ∈ P by auto

from ⟨X ≠ Y⟩ X′ Y′ have X′ ≠ Y′ by auto
from this X′ Y′ have X′ ∩ Y′ = {} using (partition-on B P) by (auto elim!: partition-onE)
from X′ Y′ have X′ ⊆ B Y′ ⊆ B using (partition-on B P) by (auto elim!: partition-onE)
from this ⟨X′ ∩ Y′ = {}⟩ X′ Y′ ⟨X′ ≠ Y′⟩ show X ∩ Y = {} using auto

qed

lemma partition-on-map:
assumes inj-on f A
assumes partition-on A P
shows partition-on (f ◦ A) (op ◦ f ◦ P)
proof –
{ fix X Y
  assume X ∈ P Y ∈ P f ◦ X ≠ f ◦ Y
  moreover from assms have ∨ p ∈ P. ∨ p' ∈ P. p ≠ p' → p ∩ p' = {} and inj-on f (∪ P)
  by (auto elim!: partition-onE)
  ultimately have f ◦ X ∩ f ◦ Y = {} unfolding inj-on-def by auto (metis IntI empty-iff rev-image-eqI+)
}
from assms this show partition-on (f ◦ A) (op ◦ f ◦ P)
by (auto intro!: partition-onI elim!: partition-onE) qed

lemma set-of-partition-on-map:
assumes inj-on f A
shows (op ◦ (op ◦ f) ◦ { P. partition-on A P}) = { P. partition-on (f ◦ A) P}
proof (rule set-eqI')
fix x
assume x ∈ (op ◦ (op ◦ 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 A X ∈ P → x ∈ X → x ∈ f ◦ A
  by (auto elim!: partition-onE)
have (op ◦ (f ◦ the-inv-into A f)) ◦ P = (op ◦ f ◦ op ◦ (the-inv-into A f) ◦ P)
  by (simp add: image-comp comp-image)


moreover have $P = \text{op } (f \circ \text{the-inv-into } A \ f) \ ' P$
proof (rule set-eqI')
  fix $X$
  assume $X: X \in P$
moreover from $X \text{ mem have in-range: } \forall x \in X, x \in f \ ' A$ by auto
moreover have $X = (f \circ \text{the-inv-into } A \ f) \ ' X$
proof (rule set-eqI')
  fix $x$
  assume $x \in X$
  show $x \in (f \circ \text{the-inv-into } A \ f) \ ' X$
  proof (rule image-eqI)
    from in-range $\langle x \in X \rangle$ assms
    show $x = (f \circ \text{the-inv-into } A \ f) \ ' x$
    proof (auto simp add $f\text{-the-inv-into-f}$)
      from $x \in X$ show $x \in X$ by assumption
    qed
  qed
next
  fix $x$
  assume $x \in (f \circ \text{the-inv-into } A \ f) \ ' X$
  from this obtain $x'$ where $x', x' \in X \land x = f (\text{the-inv-into } A \ f x')$ by auto
  from in-range $x'$ have $f: f (\text{the-inv-into } A \ f x') \in X$
    by (auto intro: $\langle \text{inj-on } f A \rangle$
  from $x' \langle X \in P \rangle$ show $x \in X$ by auto
  qed
ultimately show $X \in \text{op } (f \circ \text{the-inv-into } A \ f) \ ' P$ by auto
next
  fix $X$
  assume $X \in \text{op } (f \circ \text{the-inv-into } A \ f) \ ' P$
moreover
  \{\n    fix $Y$
    assume $Y \in P$
    from this $\langle \text{inj-on } f A \rangle$ mem have $\forall x \in Y, f (\text{the-inv-into } A \ f x) = x$
      by (auto simp add $f\text{-the-inv-into-f}$)
    from this have $(f \circ \text{the-inv-into } A \ f) \ ' Y = Y$ by force
  \}
  ultimately show $X \in P$ by auto
qed
ultimately have $P: P = \text{op } f \ ' \text{op } (\text{the-inv-into } A \ f) \ ' P$ by simp
have $A\text{-eq}: A = \text{the-inv-into } A \ f \ ' f \ ' A$ by (simp add: assms)
from $\langle \text{inj-on } f A \rangle$ have $\text{inj-on } (\text{the-inv-into } A \ f) (f ' A)$
  using $\langle \text{partition-on } (f ' A) P \rangle$ by (simp add: inj-on-the-inv-into)
from this have $\text{op } (\text{the-inv-into } A \ f) \ ' P \in \{P, \text{partition-on } A P\}$
  using $\langle \text{partition-on } (f ' A) P \rangle$ by (subst $A\text{-eq}$, auto intro: $\langle \text{partition-on } \text{map} \rangle$
from $P$ this show $P \in \text{op } (\text{op } f) \ ' \{P, \text{partition-on } A P\}$ by (rule image-eqI)
qed
end
2 Bell Numbers and Spivey’s Generalized Recurrence

theory Bell-Numbers
imports
  HOL-Library.FuncSet
  HOL-Library.Monad-Syntax
  HOL-Library.Stirling
  Card-Number-Partitions.Additions-to-Main
  Set-Partition
  HOL-Eisbach.Eisbach
begin

2.1 Preliminaries

2.1.1 Additions to FuncSet

lemma extensional-funcset-ext:
  assumes f ∈ A → E B g ∈ A → E B
  assumes ∀ x. x ∈ A =⇒ f x = g x
  shows f = g
using assms by (metis PiE-iff extensionalityI)

2.1.2 Additions for Injectivity Proofs

lemma inj-on-impl-inj-on-image:
  assumes inj-on f A
  assumes ∀ x ∈ X =⇒ x ⊆ A
  shows inj-on (op ' f) X
using assms by (meson inj-onI inj-on-image-eq-iff)

lemma injectivity-union:
  assumes A ∪ B = C ∪ D
  assumes P A P C
  assumes Q B Q D
  assumes ∀ S T. P S =⇒ Q T =⇒ S ∩ T = {}
  shows A = C ∧ B = D
using assms Int-Un-distrib Int-commute inf-sup-absorb by blast+

lemma injectivity-image:
  assumes f ' A = g ' A
  assumes ∀ x ∈ A. invert (f x) = x ∧ invert (g x) = x
  shows ∀ x ∈ A. f x = g x
using assms by (metis (no-types, lifting) image-iff)

lemma injectivity-image-union:
  assumes (λX. X ∪ F X) ' P = (λX. X ∪ G X) ' P'
  assumes ∀ X ∈ P. X ⊆ A ∀ X ∈ P'. X ⊆ A
  assumes ∀ X ∈ P. ∀ y ∈ F X. y ∉ A ∀ X ∈ P'. ∀ y ∈ G X. y ∉ A
  shows P = P'}
proof
  show $P \subseteq P'$
  proof
    fix $X$
    assume $X \in P$
    from assms(1) this obtain $X'$ where $X' \in P'$ and $X \cup F X = X' \cup G X'$
    by (metis imageE image-eqI)
    moreover from assms(2,4) have $X: (X \cup F X) \cap A = X$ by auto
    moreover from assms(3,5) have $X': (X' \cup G X') \cap A = X'$ by auto
    ultimately have $X = X'$ by simp
    from this $\langle X' \in P' \rangle$ show $X \in P'$ by auto
  qed
next
  show $P' \subseteq P$
  proof
    fix $X'$
    assume $X' \in P'$
    from assms(1) this obtain $X$ where $X \in P$ and $X \cup F X = X' \cup G X'$
    by (metis imageE image-eqI)
    moreover from assms(2,4) have $X: (X \cup F X) \cap A = X$ by auto
    moreover from assms(3,5) have $X': (X' \cup G X') \cap A = X'$ by auto
    ultimately have $X = X'$ by simp
    from this $\langle X \in P \rangle$ show $X' \in P$ by auto
  qed
qed

2.1.3 Disjointness under Function Application

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

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

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

lemma disjoint-bind': $\forall S T f g. (\forall s t. s \in S \land t \in T \Rightarrow f s \cap g t = \{\}) \Rightarrow (S \gg f) \cap (T \gg g) = \{\}$
by fastforce

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

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

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

\subsection{2.1.4 Cardinality Theorems for Set.bind}

lemma \text{finite-bind}:
assumes \(\text{finite } S\)
assumes \(\forall x \in S. \text{finite } (f\, x)\)
shows \(\text{finite } (S \gg f)\)
using \(\text{assms by (simp add: bind-UNION)}\)

lemma \text{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 \(\text{card } (S \gg f) = \text{card } (\bigcup f\, S)\)
  by (simp add: bind-UNION)
also have \(\text{card } (\bigcup f\, S) = \sum_{x \in S. \text{card } (f\, x)}\)
  using \(\text{assms unfolding disjoint-family-on-def}\)
  by (subst card-Union-image) simp+
finally show \(?\text{thesis} .\)
qed

lemma \text{card-bind-constant}:
assumes \(\text{finite } S\)
assumes \(\forall X \in S. \text{finite } (f\, X)\)
assumes \(\text{disjoint-family-on } f\, S\)
assumes \(\forall x. x \in S \implies \text{card } (f\, x) = k\)
shows \(\text{card } (S \gg f) = \text{card } S \ast k\)
using \(\text{assms by (simp add: card-bind)}\)
2.2 Definition of Bell Numbers

definition Bell :: nat ⇒ nat
where
Bell n = card {P. partition-on {0..<n} P}

Show that definition holds for any set A with cardinality n

lemma Bell-altdef:
assumes finite A
shows Bell (card A) = card {P. partition-on A P}

proof –
from ⟨finite A⟩ obtain f where bij: bij-betw {0..<card A} A
using ex-bij-betw-nat-finite by blast
from this have inj: inj-on f {0..<card A}
using bij-betw-imp-inj-on by blast
from bij have image-f-eq: A = f ′ {0..<card A}
using bij-betw-imp-surj-on by blast
have inj-on (op ′ (op ′ f)) {P. partition-on 0..<card A} =
{P. partition-on A P}
by (auto elim: partition-onE)
ultimately have bij-betw (op ′ (op ′ f)) {P. partition-on 0..<card A} P
{P. partition-on A P}
by (auto intro: bij-betw-imageI)
from this ⟨finite A⟩ show ?thesis
unfolding Bell-def
by (subt bij-betw-iff-card[symmetric]) (auto intro: finitely-many-partition-on)
qed

lemma Bell-0:
Bell 0 = 1
by (auto simp add: Bell-def partition-on-empty)

2.3 Construction of the Partitions

definition construct-partition-on :: 'a set ⇒ 'a set ⇒ 'a set set
where
construct-partition-on B C =
do {  
k ← {0..card B};
j ← {0..card C};
P ← {P. partition-on C P ∧ card P = j};
B' ← {B'. B ′ ⊆ B ∧ card B ′ = k};
Q ← {Q. partition-on B ′ Q};
f ← (B - B ′) → E P;
P ′ ← {(λX. X ∪ {x ∈ B - B ′. f x = X}) ′ P};
\{P' \cup Q\}

**Lemma** construct-partition-on:
- **Assumes** finite B finite C
- **Assumes** $B \cap C = \{\}$
- **Shows** construct-partition-on $B C = \{P. \text{partition-on} (B \cup C) P\}$

**Proof** (rule set-eqI)
- **Fix** $Q'$
  - **Assume** $Q' \in \text{construct-partition-on} B C$
  - **From this obtain** $j k P P' Q B' f$
    - **Where** $j \leq \text{card} C$
    - **And** $k \leq \text{card} B$
    - **And** $P$: partition-on $C P \land \text{card} P = j$
    - **And** $B': B' \subseteq B \land \text{card} B' = k$
    - **And** $Q$: partition-on $B' Q$
    - **And** $f: f \in B - B' \rightarrow P$
    - **And** $P': P' = (\lambda X. X \cup \{x \in B - B'. f x = X\}) \triangledown P$
    - **And** $Q': Q' = P' \cup Q$
  - **Unfolding** construct-partition-on-def by auto
  - **From** $P f$ **have** partition-on $(B - B' \cup C) P'$
    - **Unfolding** $P'$ using $(B \cap C = \{\})$
      - **By** (intro partition-on-insert-elements) auto
  - **From** this **have** partition-on $((B - B' \cup C) \cup B') Q'$
    - **Unfolding** $Q'$ using $(B' \cup B \cap C = \{\})$ by (auto intro: partition-on-union)
  - **From** this **have** partition-on $(B \cup C) Q'$
    - **Using** $B'$ by (metis Diff-partition sup.assoc sup.commute)
  - **From** this **show** $Q' \in \{P. \text{partition-on} (B \cup C) P\}$ by auto
- **Next**
  - **Fix** $Q$
    - **Assume** $Q': Q' \in \{Q'. \text{partition-on} (B \cup C) Q'\}$
    - **From** $Q'$ **have** $\{} \notin Q'$ by (auto elim!: partition-onE)
    - **Obtain** $Q$ where $Q$: $Q = (\langle \lambda X. \text{if } X \subseteq B \text{ then } X \text{ else } \{\} \rangle \triangledown Q') - \{\{\}\}$ by blast
    - **Obtain** $P'$ where $P'$: $P' = (\langle \lambda X. \text{if } X \subseteq B \text{ then } X \text{ else } \{\} \rangle \triangledown Q') - \{\{\}\}$ by blast
  - **From** $P' Q \{\{} \notin Q'$ **have** $Q'$-prop: $Q' = P' \cup Q$ by auto
  - **Have** $P'$-nosubset: $\forall X \in P', \neg X \subseteq B$
    - **Unfolding** $P'$ by auto
  - **Moreover** have $\forall X \in P', X \subseteq B \cup C$
    - **Using** $Q' P'$ by (auto elim: partition-onE)
  - **Ultimately** have $P'$-witness: $\forall X \in P', \exists x. x \in X \cap C$
    - **Using** $B \cap C = \{\}$ by fastforce
  - **Obtain** $B'$ where $B'$: $B' = \bigcup Q$ by blast
    - **Have** $Q$-prop: partition-on $B' Q$
      - **Using** $B' Q' Q'$-prop partition-on-split2 mem-Collect-eq by blast
  - **Have** $\bigcup P' = B - B' \cup C$
    - **Proof**
      - **Have** $\bigcup Q' = B \cup C \forall X \in Q'. \forall X' \in Q'. X \neq X' \rightarrow X \cap X' = \{\}$
        - **Using** $Q'$ unfolding partition-on-def disjoint-def by auto

14
from this show $\bigcup P' \subseteq B - B' \cup C$

unfolding $P' B' Q$ by auto blast

next

show $B - B' \cup C \subseteq \bigcup P'$

proof

fix $x$
assume $x \in B - B' \cup C$
from this obtain $X$ where $x \in X X \in Q'$
using $Q'$ by (metis Diff-iff Un-iff mem-Collect-eq partition-on-partition-on-unique)

have $\forall X \in Q', X \subseteq B \rightarrow X \subseteq B'$ unfolding $B' Q$ by auto
from this $x \in B - B' \cup C$ have $\neg X \subseteq B$
using $\langle B \cap C = \{\} \rangle$ by auto
from this $\langle x \in X \rangle$ have $X \in P'$ using $P'$ by auto
from this $\langle x \in X \rangle$ show $x \in \bigcup P'$ by auto

qed

qed

from this have partition-on-P': partition-on $(B - B' \cup C) P'$
using partition-on-split1 $Q'$-prop $Q'$ mem-Collect-eq by fastforce

obtain $P$ where $P: P = (\lambda X. X \cap C) \cdot P'$ by blast
from $P$ partition-on-P' $P'$-witness have partition-on $C P$
using partition-on-intersect-on-elements by auto

obtain $f$ where $f: f = (\lambda x. if x \in B - B' \cup C then (THE X. x \in X \land X \in P') \cap C else undefined) by blast$

have $P'$-prop: $P' = (\lambda X. X \cup \{x \in B - B'. f x = X\}) \cdot P'$

proof

{ fix $X$
assume $X \in P'$

have $X$-subset: $X \subseteq (B - B') \cup C$
using partition-on-P' $\langle X \in P' \rangle$ by (auto elim: partition-onE)

have $X = X \cap C \cup \{x \in B - B'. f x = X \cap C\}$

proof

{ fix $x$
assume $x \in X$

from this $X$-subset have $x \in (B - B') \cup C$ by auto
from this have $x \in X \cap C \cup \{xa \in B - B'. f xa = X \cap C\}$

proof

assume $x \in C$
from this $\langle x \in X \rangle$ show $\neg$thesis by simp

next
assume $x \in B - B'$

from partition-on-P' $\langle x \in X \rangle \langle X \in P' \rangle$ have $\neg (THE X. x \in X \land X \in P') = X$

by (simp add: partition-on-the-part-eq)
from $\langle x \in B - B' \rangle$ this show $\neg$thesis unfolding $f$ by auto

qed

}
from this show \( X \subseteq X \cap C \cup \{x \in B - B'. f x = X \cap C\} \) by auto

next
show \( X \cap C \cup \{xa \in B - B'. f xa = X \cap C\} \subseteq X \)
proof
fix \( x \)
assume \( x \in X \cap C \cup \{x \in B - B'. f x = X \cap C\} \)
from this show \( x \in X \)
proof
assume \( x \in X \cap C \)
from this show ?thesis by simp
next
assume \( x \in \{x \in B - B'. f x = X \cap C\} \)
from this obtain \( y \) where \( y \in X \cap C \) by auto
from this eq have \( y \in \{x \in X \cap X \in P'\} \cap C \) by simp
from \( y \) have \( y \in \{x \in X \cap X \in P'\} \) by auto
moreover have \( \exists! X. y \in X \cap X \in P' \)
  using partition-on-P' by (auto intro: partition-on-partition-on-unique)
moreover have \( \{x \in B - B'. f x = X \cap C\} = \{x \in B - B'. x \in X\} \)
proof
{ fix \( x \)
assume \( x \in B - B' \)
from this have \( \exists! X. x \in X \land X \in P' \)
}
using partition-on-\(P\)\(^{\prime}\) by (auto intro!: partition-on-partition-on-unique)

from this have in-p: (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) \(\in P\)

and x-in: \(x \in (\text{THE } X. \ x \in X \land X \in P)\)

by (metis (mono-tags, lifting) theI)+

have \(f x = X \cap C \iff (\text{THE } X. \ x \in X \land X \in P') \cap C = X \cap C\)

using (\(x \in B - B'\) unfolding \(f\) by auto

also have \(... \iff (\text{THE } X. \ x \in X \land X \in P') = X\)

proof

assume (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) = \(X\)

from this show (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) \(\cap C = X \cap C\) by auto

next

assume (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) \(\cap C = X \cap C\)

have (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) \(\cap X \neq \{\}\)

using P'-witness (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) \(\cap C = X \cap C\) \(\cap X \in P\)\(^{\prime}\) by fastforce

from this show (THE \(X\). \(x \in X \land X \in P\)\(^{\prime}\) = \(X\)

using partition-on-\(P\)\(^{\prime}\) [unfolded partition-on-def disjoint-def] in-p \(\langle X \in P\)\(^{\prime}\) by metis

qed

also have \(... \iff x \in X\)

using ex1 \(\langle X \in P\)\(^{\prime}\) x-in by (auto; metis (no-types, lifting) the-equality)

finally have \(f x = X \cap C \iff x \in X\).

} from this show \(\text{thesis}\) by auto

qed

moreover have \(X \subseteq B - B' \cup C\)

using partition-on-\(P\)\(^{\prime}\) \(\langle X \in P\)\(^{\prime}\) by (blast elim: partition-onE)

ultimately have \(X \cap C \cup \{x \in B. \ x \notin B' \land f x = X \cap C\} = X\) by auto

} from this x-in-image have \(x \in P\)\(^{\prime}\) unfolding \(P\) by auto

} from this show \((\lambda X. \ X \cup \{x \in B - B'. \ f x = X\}) \cdot P \subseteq P\)\(^{\prime}\).

qed

from partition-on-\(P\)\(^{\prime}\) have \(f \cdot \text{prop}: \ f \in (B - B') \rightarrow E \ P\)

unfolding \(f \cdot \text{prop}\) by (auto simp add: partition-on-the-part-mem)

from \(Q\) \(B'\) have \(B' \subseteq B\) by auto

obtain \(k\) where \(k: k = \text{card } B'\) by blast

from \(\text{finite } B\) \(\langle B' \subseteq B\); \(k\) \text{ have } \text{prop}: \ k \in \{0...\text{card } B\}\) by (simp add: card-mono)

obtain \(j\) where \(j: j = \text{card } P\) by blast

from \(j\) \(\langle \text{partition-on } C\ P\); \(\text{have } \text{prop}: \ j \in \{0...\text{card } C\}\)

by (simp add: assms(2) partition-on-le-set-elements)

from \(\text{partition-on } C \ P\); \(\text{have } \text{prop}: \ P\) \(\text{prop}\) \(\text{prop}\) \(\text{prop}\) \(\text{prop}\) \(\text{prop}\) \(\text{prop}\)

by (auto simp del: atLeastAtMost-iff) blast

qed
2.4 Injectivity of the Set Construction

lemma injectivity:

assumes $B \cap C = \{\}$
assumes $P$: $(\text{partition-on } C \land \text{card } P = j) \land (\text{partition-on } C \land \text{card } P' = j')$
assumes $B'': (B' \subseteq B \land \text{card } B' = k) \land (B'' \subseteq B \land \text{card } B'' = k')$
assumes $Q$: partition-on $B' Q \land \text{partition-on } B'' Q'$
assumes $f$: $f \in B - B' \rightarrow E P \land g \in B - B'' \rightarrow E P''$
assumes $P'': P'' \in \{(\lambda X. X \cup \{x \in B - B', f x = X\} \cdot P) \land P'' \in \{(\lambda X. X \cup \{x \in B - B'', g x = X\} \cdot P')\}$
assumes eq-result: $P'' \cup Q = P''' \cup Q'$
shows $f = g$ and $Q = Q'$ and $B' = B''$
and $P = P'$ and $j = j'$ and $k = k'$

proof

have $P$-nonempty-sets: $\forall X \in P. \exists c \in C. c \in X \land X \in P' \land \exists c \in C. c \in X$
using $P$ by (force elim: partition-onE)+
have 1: $\forall X \in P'''. \exists c \in C. c \in X \land X \in P''' \land \exists c \in C. c \in X$
using $P''$ $P$-nonempty-sets by fastforce+
have 2: $\forall X \in Q. \forall x \in X. x \not\in C \land X \in Q'. \forall x \in X. x \not\in C$
using $(B \cap C = \{\}) \cdot Q \land B' \land B''$ by (auto elim: partition-onE)
from eq-result have $P'' = P'''$ and $Q = Q'$
by (auto dest: injectivity-union[OF - 1 2])
from this $Q$ show $Q = Q'$ and $B' = B''$
by (auto intro!: partition-on-E-implies-eq-carrier)
have subset-C: $\forall X \in P. X \subseteq C \land X \in P' \land X \subseteq C$
using $P$ by (auto elim: partition-onE)
have eq-image: $(\lambda X. X \cup \{x \in B - B', f x = X\} \cdot P = (\lambda X. X \cup \{x \in B - B'', g x = X\} \cdot P')$
using $P' \cdot P'' = P'''$ by auto
from this $(B \cap C = \{\})$ show $P = P'$
by (auto dest: injectivity-image-union[OF - subset-C])
have eq2: $(\lambda X. X \cup \{x \in B - B', f x = X\} \cdot P = (\lambda X. X \cup \{x \in B - B', g x = X\} \cdot P)$
using $(P = P') \cdot B' = B''$ eq-image by simp
from $P$ have $P$-props: $\forall X \in P. X \subseteq C \land X \in P. X \neq \{\}$ by (auto elim: partition-onE)
have invert: $\forall X \in P. (X \cup \{x \in B - B', f x = X\}) \cap C = X \land (X \cup \{x \in B - B', g x = X\}) \cap C = X$
using $(B \cap C = \{\})$ P-props by auto
have eq3: $\forall X \in P. (X \cup \{x \in B - B', f x = X\}) = (X \cup \{x \in B - B', g x = X\})$
using injectivity-image[OF eq2 invert] by blast
have eq4: $\forall X \in P. \{x \in B - B', f x = X\} = \{x \in B - B', g x = X\}$
proof
fix $X$
assume $X \in P$
from this $P$ have $X \subseteq C$ by (auto elim: partition-onE)
have disjoint: $X \cap \{x \in B - B', f x = X\} = \{\} \land X \cap \{x \in B - B', g x = X\} = \{\}$
18
using \( B \cap C = \{\} \); \( X \subseteq C \) by auto 

from eq3 \( X \in P \) have \( X \cup \{ x \in B - B', f x = X \} = X \cup \{ x \in B - B', g x = X \} \) by auto 

from this disjoint show \( \{ x \in B - B', f x = X \} = \{ x \in B - B', g x = X \} \) by (auto intro: injectivity-union) 

qed 

from eq4 f have eq5: \( \forall b \in B - B'. f b = g b \) by blast 

from eq5 f (\( B' = B'' \) \( P = P' \)) show eq6: \( f = g \) by (auto intro: extensional-funcset-ext) 

qed 

2.5 The Generalized Bell Recurrence Relation 

**Theorem Bell-eq:**

\[
\text{Bell} \ (n + m) = (\sum k \leq n. \sum j \leq m. \ j \ hat (n - k) \ * \ \text{Stirling m j} \ * \ (n \ choose \ k) \ * \ \text{Bell k})
\]

**Proof**

- def A ::= \{0..<n + m\} 
- def B ::= \{0..<n\} 
- def C ::= \{n..<n + m\} 

have A = B \cup C B \cap C = \{\} finite B card B = n finite C card C = m 

unfolding A-def B-def C-def by auto 

have step1: \( \text{Bell} \ (n + m) = \text{card} \ \{ P. \ \text{partition-on} A P \} \) 

unfolding Bell-def A-def .. 

from \( \langle A = B \cup C \rangle \langle B \cap C = \{\} \rangle \langle \text{finite} B \rangle \langle \text{finite} C \rangle \) 

have step2: \( \text{card} \ \{ P. \ \text{partition-on} A P \} = \text{card} \ \{ \text{construct-partition-on} B C \} \) 

by (simp add: construct-partition-on) 

note injectivity = injectivity[OF \( B \cap C = \{\} \)] 

let \( expr = \) do 

\( k \leftarrow \{0..\text{card} B\} \); 
\( j \leftarrow \{0..\text{card} C\} \); 
\( P \leftarrow \{ P. \ \text{partition-on} C P \text{\ and} \ \text{card} P = j \} \); 
\( B' \leftarrow \{ B'. B' \subseteq B \text{\ and} \ \text{card} B' = k \} \); 
\( Q \leftarrow \{ Q. \ \text{partition-on} B' Q \} \); 
\( f \leftarrow (B - B') \rightarrow E P \); 
\( P' \leftarrow \{ (\lambda X. X \cup \{ x \in B - B', f x = X \}) \ ' P \} \); 
\( \{ P' \cup Q \} \) 

let \( S \gg expr = \) expr 

\{ 
  fix k 
  assume k: k \in \{..\text{card} B\} 
  let expr = \( \text{comp} = \) expr 
  \{ 
    fix j 
    assume j \in \{..\text{card} C\} 
    let expr = \( \text{comp} = \) expr 
  \} 
\}
let ?S ≔ ?comp = ?expr
from (finite C) have finite ?S
by (intro finite-Collect-conjI disjI1 finitely-many-partition-on)

{ fix P
assume P: P ∈ \{P, partition-on C P \land card P = j\}
from this have partition-on C P by simp
let ?expr = ?comp P
let ?S ≔ ?comp = ?expr
have finite P
using P ⟨finite C⟩ by (auto intro: finite-elements)
from (finite B) have finite ?S by (auto simp add: finite-subset)
moreover

{ fix B'
assume B': B' ∈ \{B', B' ⊆ B \land card B' = k\}
from this have B' ⊆ B by simp
let ?expr = ?comp B'
let ?S ≔ ?comp = ?expr
from (finite B) have finite B'
using B' by (auto simp add: finite-subset)
from (finite B') have finite \{Q, partition-on B' Q\}
by (rule finitely-many-partition-on)
moreover

{ fix Q
assume Q: Q ∈ \{Q, partition-on B' Q\}
let ?expr = ?comp Q
let ?S ≔ ?comp = ?expr

{ fix f
assume f ∈ B − B' → E P
let ?expr = ?comp f
let ?S ≔ ?comp = ?expr
have disjoint-family-on ?comp ?S
by (auto intro: disjoint-family-onI)
from this have card ?expr = 1
by (simp add: card-bind-constant)
moreover have finite ?expr
by (simp add: finite-bind)
ultimately have finite ?expr ∧ card ?expr = 1 by blast
}
moreover have finite ?S
using (finite B) (finite P) by (auto intro: finite-PiE)
moreover have disjoint-family-on ?comp ?S
using P B' Q
by (injectivity-solver rule: local.injectivity(1))
moreover have card ?S = j ^ (n − k)
proof –
have \( \text{card} (B - B') = n - k \)

using \( B' \langle \text{finite} B' \rangle \langle \text{card} B = n \rangle \)
by (subst card-Diff-subset) auto

from this show \(?\text{thesis}\)

using \( \langle \text{finite} B \rangle \) \( P \)
by (subst card-PiE) (simp add: prod-constant)

qed

ultimately have \( \text{card} \ ?\text{expr} = j \cdot (n - k) \)
by (simp add: card-bind-constant)

moreover have \( \text{finite} \ ?\text{expr} \)
using \( \langle \text{finite} \ ?S \rangle \langle \text{finite} \ \{ P, \ \text{partition-on} \ P \land \text{card} P = j \} \rangle \)
by (auto intro: finite-bind)

ultimately have \( \text{finite} \ ?\text{expr} \land \text{card} \ ?\text{expr} = j \cdot (n - k) \) by blast

} note inner = this

moreover have \( \text{card} \ ?S = \text{Bell} k \)
using \( B' \langle \text{finite} B' \rangle \) by (auto simp add: Bell-altdef[symmetric])

moreover have \( \text{disjoint-family-on} \ ?\text{comp} \ ?S \)
using \( P \ B' \)
by (injectivity-solver rule: local.injectivity(2))

ultimately have \( \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot \text{Bell} k \)
by (subst card-bind-constant) auto

moreover have \( \text{finite} \ ?\text{expr} \)
using inner \( \langle \text{finite} \ ?S \rangle \) by (auto intro: finite-bind)

ultimately have \( \text{finite} \ ?\text{expr} \land \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot \text{Bell} k \) by blast

} note inner = this

moreover have \( \text{card} \ ?S = \text{n choose} k \)
using \( \langle \text{card} B = n \rangle \langle \text{finite} B \rangle \) by (simp add: n-subsets)

moreover have \( \text{disjoint-family-on} \ ?\text{comp} \ ?S \)
using \( P \)
by (injectivity-solver rule: local.injectivity(3))

ultimately have \( \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot (\text{n choose} k) \cdot \text{Bell} k \)
by (subst card-bind-constant) auto

moreover have \( \text{finite} \ ?\text{expr} \)
using inner \( \langle \text{finite} \ ?S \rangle \) by (auto intro: finite-bind)

ultimately have \( \text{finite} \ ?\text{expr} \land \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot (\text{n choose} k) \cdot \text{Bell} k \) by blast

} note inner = this

moreover note \( \langle \text{finite} \ ?S \rangle \)
moreover have \( \text{card} \ ?S = \text{Stirling} m j \)
using \( \langle \text{finite} C \rangle \langle \text{card} C = m \rangle \) by (simp add: card-partition-on)

moreover have \( \text{disjoint-family-on} \ ?\text{comp} \ ?S \)
by (injectivity-solver rule: local.injectivity(4))

ultimately have \( \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot \text{Stirling} m j \cdot (\text{n choose} k) \cdot \text{Bell} k \)
by (subst card-bind-constant) auto

moreover have \( \text{finite} \ ?\text{expr} \)
using inner \( \langle \text{finite} \ ?S \rangle \) by (auto intro: finite-bind)

ultimately have \( \text{finite} \ ?\text{expr} \land \text{card} \ ?\text{expr} = j \cdot (n - k) \cdot \text{Stirling} m j \cdot (n \cdot \text{Bell} k) \) by blast
choose k) * Bell k by blast

    } note inner = this
moreover have finite ?S by simp
moreover have disjoint-family-on ?comp ?S
    by (injectivity-solver rule: local.injectivity(5))
ultimately have card ?expr = (∑ j≤m. j *(n − k) * Stirling m j * (n choose
k) * Bell k) (is - = ?formula)
    using (card C = m) by (subst card-bind) (auto intro: sum.cong)
moreover have finite ?expr
    using inner (finite ?S) by (auto intro: finite-bind)
ultimately have finite ?expr ∧ card ?expr = ?formula by blast

moreover have finite ?S by simp
moreover have disjoint-family-on ?comp ?S
    by (injectivity-solver rule: local.injectivity(6))
ultimately have step3: card (construct-partition-on B C) = (∑ k≤n. ∑ j≤m. j *(n − k) * Stirling m j * (n choose k) * Bell k)

    unfolding construct-partition-on-def
    using (card B = n) by (subst card-bind) (auto intro: sum.cong)
from step1 step2 step3 show ?thesis by auto
qed

2.6 Corollaries of the Generalized Bell Recurrence

corollary Bell-Stirling-eq:
Bell m = (∑ j≤m. Stirling m j)

proof –
    have Bell m = Bell (0 + m) by simp
    also have ... = (∑ j≤m. Stirling m j)
        unfolding Bell-eq[0] by (simp add: Bell-0)
    finally show ?thesis .
qed

corollary Bell-recursive-eq:
Bell (n + 1) = (∑ k≤n. (n choose k) * Bell k)
unfolding Bell-eq[0 - 1] by simp

end

References

[1] N. J. A. Sloane. A000110: Bell or exponential numbers: number of ways
to partition a set of n labeled elements. In The On-Line Encyclopedia of

of Integer Sequences, 11, 2008. Electronic copy available at