Spivey’s Generalized Recurrence for Bell Numbers

Lukas Bulwahn

October 27, 2022

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 Bell Numbers and Spivey’s Generalized Recurrence

1.1 Preliminaries .................................................. 2
1.1.1 Additions to FuncSet .................................... 2
1.1.2 Additions for Injectivity Proofs ......................... 2
1.2 Definition of Bell Numbers ................................. 3
1.3 Construction of the Partitions ............................. 4
1.4 Injectivity of the Set Construction ....................... 8
1.5 The Generalized Bell Recurrence Relation ............... 9
1.6 Corollaries of the Generalized Bell Recurrence .......... 12

1 Bell Numbers and Spivey’s Generalized Recurrence

theory Bell-Numbers
imports
  HOL-Library.FuncSet
  HOL-Library.Monad-Syntax
1.1 Preliminaries

1.1.1 Additions to FuncSet

\textbf{lemma} \textit{extensional-funcset-ext}:
\begin{enumerate}
\item \textbf{assumes} \( f \in A \rightarrow E \) \( g \in A \rightarrow E \)
\item \textbf{assumes} \( \forall x \in A \Rightarrow f x = g x \)
\item \textbf{shows} \( f = g \)
\end{enumerate}
\textbf{using} \textit{assms} by \textit{(metis PiE-iff extensionalityI)}

1.1.2 Additions for Injectivity Proofs

\textbf{lemma} \textit{inj-on-impl-inj-on-image}:
\begin{enumerate}
\item \textbf{assumes} \( \text{inj-on} \ f A \)
\item \textbf{assumes} \( \forall x \in X \Rightarrow x \subseteq A \)
\item \textbf{shows} \( \text{inj-on} \ (\langle \rangle \ f) X \)
\end{enumerate}
\textbf{using} \textit{assms} by \textit{(meson inj-onI inj-on-image-eq-iff)}

\textbf{lemma} \textit{injectivity-union}:
\begin{enumerate}
\item \textbf{assumes} \( A \cup B = C \cup D \)
\item \textbf{assumes} \( \forall X \in P \). \( \forall y \in F X. \ y \notin A \)
\item \textbf{shows} \( \forall X \in P'. \ y \notin G X. \ y \notin A \)
\end{enumerate}
\textbf{shows} \( A = C \land B = D \)
\textbf{using} \textit{assms} \textit{Int-Un-distrib Int-commute inf-sup-absorb by blast+}

\textbf{lemma} \textit{injectivity-image}:
\begin{enumerate}
\item \textbf{assumes} \( f ' A = g ' A \)
\item \textbf{assumes} \( \forall x \in A. \ \text{invert} (f x) = x \land \text{invert} (g x) = x \)
\item \textbf{shows} \( \forall x \in A. \ f x = g x \)
\end{enumerate}
\textbf{using} \textit{assms} by \textit{(metis (no-types, lifting) image-iff)}

\textbf{lemma} \textit{injectivity-image-union}:
\begin{enumerate}
\item \textbf{assumes} \( \forall X \in P. \ X \subseteq A \land X \in P'. \ X \subseteq A \)
\item \textbf{assumes} \( \forall X \in P. \ \forall y \in F X. \ y \notin A \land X \in P'. \ \forall y \in G X. \ y \notin A \)
\item \textbf{shows} \( P = P' \)
\end{enumerate}
\textbf{proof}
\begin{enumerate}
\item \textbf{show} \( P \subseteq P' \)
\item \textbf{proof}
\item \textbf{fix} \( X \)
\item \textbf{assume} \( X \in P \)
\item \textbf{from} \textit{assms}(1) \textbf{this} \textbf{obtain} \( X' \) \textbf{where} \( X' \in P' \) \textbf{and} \( X \cup F X = X' \cup G X' \)
\item \textbf{by} \textit{(metis imageE image-eqI)}
\item \textbf{moreover from} \textit{assms}(2,4) \( \langle X \in P, \ \text{have} \ X: (X \cup F X) \cap A = X \ \text{by auto} \)
moreover from assms(3,5) \( X' \in P' \) have \( X' \cap (X' \cup G X') \cap A = X' \) by auto
ultimately have \( X = X' \) by simp
from this \( X' \in P' \) 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) \( X \in P \) have \( X \cup F X \cap A = X \) by auto
moreover from assms(3,5) \( X' \in P' \) have \( X' \cap (X' \cup G X') \cap A = X' \) by auto
ultimately have \( X = X' \) by simp
from this \( X \in P \) show \( X' \in P \) by auto qed

qed

1.2 Definition of Bell Numbers

definition Bell :: nat \Rightarrow nat
where
  \( Bell n = \text{card} \ \{ P. \ \text{partition-on} \ \{0..<n\} P \} \)

lemma Bell-altdef:
  assumes finite A
  shows \( \text{Bell} (\text{card} \ A) = \text{card} \ \{ P. \ \text{partition-on} \ A \ P \} \)
proof
  from \( \text{finite} \ A \) obtain \( f \) where bij: bij-betw \( \{0..<\text{card} \ A\} \ \{\text{card} \ A\} \)
    using ex-bij-betw-nat-finite by blast
  from this have inj: inj-on \( f \ \{0..<\text{card} \ A\} \)
    using bij-betw-imp-inj-on by blast
  from bij have image-f-eq: \( A = f ' \ \{0..<\text{card} \ A\} \)
    using bij-betw-imp-surj-on by blast
  have \( \forall x \in \{ P. \ \text{partition-on} \ \{0..<\text{card} \ A\} \ P. \ \text{x} \subseteq \text{Pow} \ \{0..<\text{card} \ A\} \)
    by (auto elim: partition-onE)
  from this inj have inj-on \( f \ \{0..<\text{card} \ A\} \)
    by (intro inj-on-impl-inj-on-image[of - Pow \ \{0..<\text{card} \ A\}]\]
  moreover from inj have \( f \ \{0..<\text{card} \ A\} \)
    by (auto elim: partition-onE)
  ultimately have bij-betw \( f \ \{0..<\text{card} \ A\} \)
    by (auto intro: bij-betw-imageI)
  from this \( \text{finite} \ A \) show \( \text{thesis} \)
  unfolding Bell-def

by (subst 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)

1.3 Construction of the Partitions

definition construct-partition-on :: 'a set ⇒ 'a set ⇒ 'a set 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' ∪ Q}

lemma construct-partition-on:
  assumes finite B finite C
  assumes B ∩ C = {}
  shows construct-partition-on B C = {P. partition-on (B ∪ C) P}
proof (rule set-eqI)
  fix Q'
  assume Q' ∈ construct-partition-on B C
  from this obtain j k P P' Q B' f
    where j ≤ card C
    and k ≤ card B
    and P; partition-on C P ∧ card P = j
    and B': B' ⊆ B ∧ card B' = k
    and Q; partition-on B' Q
    and f; f ∈ B − B' →_E P
    and P': P' = (λX. X ∪ {x ∈ B − B'. f x = X}) ' P
    and Q'; Q' = P' ∪ Q
  unfolding construct-partition-on-def by auto
  from P f have partition-on (B − B' ∪ C) P'
    unfolding P' using (B ∩ C = {}),
    by (intro partition-on-insert-elements) auto
  from this Q have partition-on ((B − B' ∪ C) ∪ B') Q'
    unfolding Q' using B' (B ∩ C = {}), by (auto intro: partition-on-union)
  from this have partition-on (B ∪ C) Q'
    using B' by (metis Diff-partition sup.assoc sup.commute)
  from this show Q' ∈ {P. partition-on (B ∪ C) P} by auto
next
\textbf{fix } Q' \\
\textbf{assume } Q': Q' \in \{ Q', \text{partition-on} (B \cup C) \} \\
\textbf{from } Q' \text{ have \{} \notin Q' \text{ by } (\text{auto elim!: partition-onE}) \\
\textbf{obtain } Q \text{ where } Q: Q = ((\lambda x. \text{if } x \subseteq B \text{ then } X \text{ else } \{\}) \ ' Q') - \{\{\}\} \text{ by blast} \\
\textbf{obtain } P' \text{ where } P': P' = ((\lambda x. \text{if } x \subseteq B \text{ then } \{\} \text{ else } X) \ ' Q') - \{\{\}\} \text{ by blast} \\
\textbf{from } P' Q \\{\} \notin Q' \text{ have } Q'\text{-prop: } Q' = P' \cup Q \text{ by auto} \\
\textbf{have } P'\text{-nosubset: } \forall X \in P'. \neg X \subseteq B \\
\textbf{unfolding } P' \text{ by auto} \\
\textbf{moreover have } \forall X \in P'. X \subseteq B \cup C \\
\textbf{using } Q' \text{ by } (\text{auto elim: partition-onE}) \\
\textbf{ultimately have } P'\text{-witness: } \forall X \in P'. \exists x. x \in X \cap C \\
\textbf{using } \langle B \cap C = \{\} \rangle \text{ by fastforce} \\
\textbf{obtain } B' \text{ where } B': B' = \bigcup Q \text{ by blast} \\
\textbf{have } Q\text{-prop: } \text{partition-on } B' Q \\
\textbf{using } B' Q' \text{ by } \text{partition-on-split2 mem-Collect-eq by blast} \\
\textbf{have } \bigcup P' = B - B' \cup C \\
\textbf{proof} \\
\textbf{have } \bigcup Q' = B \cup C \forall X \in Q'. \forall X' \in Q'. X \neq X' \longrightarrow X \cap X' = \{\} \\
\textbf{using } Q' \text{ unfolding } \text{partition-on-def disjoint-def by auto} \\
\textbf{from this show } \bigcup P' \subseteq B - B' \cup C \\
\textbf{unfolding } P' B' Q \text{ by auto blast} \\
\textbf{next} \\
\textbf{show } B - B' \cup C \subseteq \bigcup P' \\
\textbf{proof} \\
\textbf{fix } x \\
\textbf{assume } x \in B - B' \cup C \\
\textbf{from this obtain } X \text{ where } X: x \in X \wedge X \in Q' \\
\textbf{using } Q' \text{ by } (\text{metis Diff-iff Un-iff mem-Collect-eq partition-on-partition-on-unique}) \\
\textbf{have } \forall X \in Q'. X \subseteq B \longrightarrow X \subseteq B' \\
\textbf{unfolding } B' Q \text{ by auto} \\
\textbf{from this } X \langle x \in B - B' \cup C \rangle \text{ have } \neg X \subseteq B \\
\textbf{using } \langle B \cap C = \{\} \rangle \text{ by auto} \\
\textbf{from this } \langle X \in Q' \rangle \text{ have } X \in P' \text{ using } P' \text{ by auto} \\
\textbf{from this } \langle x \in X \rangle \text{ show } x \in \bigcup P' \text{ by auto} \\
\textbf{qed} \\
\textbf{qed} \\
\textbf{from this have } \text{partition-on-P: } \text{partition-on } (B - B' \cup C) P' \\
\textbf{using } \text{partition-on-split1 } Q'\text{-prop } Q' \text{ mem-Collect-eq by fastforce} \\
\textbf{obtain } P \text{ where } P: P = (\lambda x. X \cap C) \ ' P' \text{ by blast} \\
\textbf{from } P \text{ partition-on-P' } P'\text{-witness have } \text{partition-on } C P \\
\textbf{using } \text{partition-on-intersect-on-elements by auto} \\
\textbf{obtain } f \text{ where } f: f = (\lambda x. \text{if } x \in B - B' \text{ then } (\text{THE } X. x \in X \wedge X \in P') \cap C \text{ else undefined}) \text{ by blast} \\
\textbf{have } P'\text{-prop: } P' = (\lambda x. X \cup \{x \in B - B', f x = X\}) \ ' P \\
\textbf{proof} \\
\{ \\
\textbf{fix } X \\
\textbf{assume } X \in P' \\
\}
have $X$-subset: $X \subseteq (B - B') \cup C$

using partition-on-$P'$ ($X \in P'$) 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 ($x \in X$) show $\mathsf{thesis}$ by simp
  next
    assume $x \in B - B'$
    from partition-on-$P'$ ($x \in X$) ($X \in P'$) have (THE $X$. $x \in X \land X \in P'$) by simp
    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 $\mathsf{thesis}$ by simp

next

assume $x$-in: $x \in \{x \in B - B'. f\, x = X \cap C\}$

from this have $\exists!X. x \in X \land X \in P'$

using partition-on-$P'$ by (auto intro: partition-on-partition-on-unique)

from $x$-in $X$-subset have eq: (THE $X$. $x \in X \land X \in P'$) $\cap C = X \cap C$

unfolding $f$ by auto

from $P'$-nosubset ($X \in P'$) have $\neg \, X \subseteq B$ by simp

from this have $X \cap C \neq \{\}$

using $X$-subset assms(3) by blast

from this obtain $y$ where $y: y \in X \cap C$ by auto

from this eq have $y$-in: $y \in (\text{THE } X. x \in X \land X \in P') \cap C$ by simp

from $y$ $y$-in have $y \in X$ $y \in (\text{THE } X. x \in X \land X \in P')$ by auto

moreover from $y$ have $\exists!X. y \in X \land X \in P'$

using partition-on-$P'$ by (simp add: partition-on-partition-on-unique)

moreover have (THE $X$. $x \in X \land X \in P'$) $\in P'$

using $\exists!X$ by (rule the1I2) auto

ultimately have (THE $X$. $x \in X \land X \in P'$) $\in X$ using $\langle X \in P'\rangle$ by auto

from this $\exists!$ show $\mathsf{thesis}$ by (auto intro: the1I2)

\end{verbatim}
\[ \text{qed} \]
\[ \text{qed} \]
\[ \text{qed} \]
\[ \text{from } x \in P' \text{ this have } X \in (\lambda X. \ X \cup \{ x \in B - B'. \ f x = X \}) \checkmark P \]
\[ \text{unfolding } P \text{ by simp} \]
\[ \} \]
\[ \text{from this show } P' \subseteq (\lambda X. \ X \cup \{ x \in B - B'. \ f x = X \}) \checkmark P \ldots \]
\[ \text{next} \]
\[ \{ \]
\[ \text{fix } x \]
\[ \text{assume } x\text{-in-image}; \ x \in (\lambda X. \ X \cup \{ x \in B - B'. \ f x = X \}) \checkmark P \]
\[ \{ \]
\[ \text{fix } X \]
\[ \text{assume } X \in P' \]
\[ \text{have } \{ x \in B - B'. \ f x = X \cap C \} = \{ x \in B - B'. \ x \in X \} \]
\[ \text{proof } \]
\[ \{ \]
\[ \text{fix } x \]
\[ \text{assume } x \in B - B' \]
\[ \text{from this have ex1: } \exists ! X. \ x \in X \land X \in P' \]
\[ \text{using partition-on-}P' \text{ by (auto intro: partition-on-partition-on-unique)} \]
\[ \text{from this have in-p: } (\text{THE } X. \ x \in X \land X \in P') \in P' \]
\[ \text{and x-in: } x \in (\text{THE } X. \ x \in X \land X \in P') \]
\[ \text{by (metis (mono-tags, lifting) theI)+} \]
\[ \text{have } f x = X \cap C \leftrightarrow (\text{THE } X. \ x \in X \land X \in P') \cap C = X \cap C \]
\[ \text{using } cx \in B - B' \text{ unfolding } f \text{ by auto} \]
\[ \text{also have ... } \leftrightarrow (\text{THE } X. \ x \in X \land X \in P') = X \]
\[ \text{proof } \]
\[ \{ \]
\[ \text{assume } (\text{THE } X. \ x \in X \land X \in P') = X \]
\[ \text{from this show } (\text{THE } X. \ x \in X \land X \in P') \cap C = X \cap C \text{ by auto} \]
\[ \text{next} \]
\[ \text{assume } (\text{THE } X. \ x \in X \land X \in P') \cap C = X \cap C \]
\[ \text{have } (\text{THE } X. \ x \in X \land X \in P') \cap X \neq \{ \} \]
\[ \text{using } P'\text{-witness } (\text{THE } X. \ x \in X \land X \in P') \cap C = X \cap C \rightarrow \{ X \in P' \text{ by metis} \]
\[ \text{qed} \]
\[ \text{also have ... } \leftrightarrow x \in X \]
\[ \text{using ex1 } x \in P' \text{-in by (auto; metis (no-types, lifting) the-equality)} \]
\[ \text{finally have } f x = X \cap C \leftrightarrow x \in X . \]
\[ \} \]
\[ \text{from this show } \{ \text{thesis by auto} \}
\[ \text{qed} \]
\[ \text{moreover have } X \subseteq B - B' \cup C \]
\[ \text{using partition-on-}P' \text{-x in } P' \text{ by (blast elim: partition-onE)} \]
\[ \text{ultimately have } X \cap C \cup \{ x \in B. \ x \notin B' \land f x = X \cap C \} = X \text{ by auto} \]
1.4 Injectivity of the Set Construction

lemma injectivity:
  assumes B ∩ C = {}
  assumes P: (partition-on C P ∧ card P = j) ∧ (partition-on C P' ∧ card P' = j')
  assumes B': (B' ⊆ B ∧ card B' = k) ∧ (B'' ⊆ B ∧ card B'' = k')
  assumes Q: partition-on B' Q ∧ partition-on B'' Q'
  assumes f: f ∈ (B - B') →_E P ∧ g ∈ (B - B'') →_E P'
  assumes P'': P'' = (λX. X ∪ {x ∈ B - B', f x = X}) · P ∧ P''' = (λX. X ∪ {x ∈ B - B'', g x = X}) · P'
  assumes eq-result: P'''' = P'''' ∧ Q'
  shows f = g and Q = Q' and B' = B''
  and P = P' and j = j' and k = k'
proof –
  have P-nonempty-sets: ∀X∈P. ∃c∈C. c ∈ X ∨ X∈P'
  using P by (force elim: partition-onE)+
  have 1: ∀X∈P''. ∃c∈C. c ∈ X ∨ X∈P'''
  using P' P-nonempty-sets by fastforce+
  have 2: ∀X∈Q. ∀x∈X. x ∉ C ∨ X∈Q'. ∀x∈X. x ∉ C
  using B ∩ C = {} ∨ Q B' by (auto elim: partition-onE)
  from eq-result have P'' = P'''' and Q = Q'
  by (auto dest: injectivity-union[OF - I 2])
  from this Q show Q = Q' and B' = B''
  by (auto intros: partition-on-eq-implies-eq-carrier)
  have subset-C: ∀X∈P. X ⊆ C ∨ X∈P'. X ⊆ C
  using P by (auto elim: partition-onE)
  have eq-image: (λX. X ∪ {x ∈ B - B', f x = X}) · P = (λX. X ∪ {x ∈ B -
1.5 The Generalized Bell Recurrence Relation

**Theorem Bell-eq:**
\[
\text{Bell} (n + m) = \left( \sum_{k \leq n} \sum_{j \leq m} j \sim (n - k) \right) * \text{Stirling} \ m \ j * (n \ \text{choose} \ k) * \text{Bell} \ k)
\]

**Proof**
- **Define:**
  - \(A\) where \(A = \{0..<n + m\}\)
  - \(B\) where \(B = \{0..<n\}\)
  - \(C\) where \(C = \{n..<n + m\}\)
- **Have:**
  - \(A = B \cup C\) \(B \cap C = \{\}\) \text{finite} \(B\) \(\text{card} \ B = n\) \text{finite} \(C\) \(\text{card} \ C = m\)
- **Unfolding:** \(A\)-def \(B\)-def \(C\)-def by auto
- **Have:**
  - **Step 1:** \(\text{Bell} (n + m) = \text{card} \ \{P. \text{partition-on} \ A \ P\}\)
  - **Unfolding:** \(\text{Bell-def} \ A\)-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: card \{P.\ partition-on A P\} = card (construct-partition-on B C)
  by (simp add: construct-partition-on)

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

let ?expr = do 
  k ← \{0..\card B\};
  j ← \{0..\card C\};
  P ← \{P.\ partition-on C P \land \card P = j\};
  B' ← \{B'.\ \card B' = k\};
  Q ← \{Q.\ partition-on B' Q\};
  f ← \(B - B') \rightarrow _E P;\n  P' ← \{(\lambda X. X \cup \{x \in B - B'.\ f x = X\}) \cdot P\};
  \\{P' \cup Q\}
}

let ?S ≥ ?comp = ?expr 
{
  fix k
  assume k: k ∈ \{..\card B\}
  let ?expr = ?comp k
  let ?S ≥ ?comp = ?expr
  
  fix j
  assume j ∈ \{..\card C\}
  let ?expr = ?comp j
  let ?S ≥ ?comp = ?expr
  from \finite C\ have finite ?S
  by (intro finite-Collect-conj 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'.\ \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
    }
  }

10
assume $Q$: $Q \in \{Q. \text{partition-on } B' \mid Q\}$
let $?expr = ?comp \ Q$
let $?S \gg \ ?comp = ?expr$
\[
\{ \\
\quad \text{fix } f \\
\quad \text{assume } f \in B - B' \to_E P \\
\quad \text{let } ?expr = ?comp \ f \\
\quad \text{let } ?S \gg \ ?comp = ?expr \\
\quad \text{have disjoint-family-on } ?comp \ ?S \\
\quad \quad \text{by } (\text{auto intro: disjoint-family-onI}) \\
\quad \quad \text{from this have card } ?expr = 1 \\
\quad \quad \text{by } (\text{simp add: card-bind-constant}) \\
\quad \quad \text{moreover have finite } ?expr \\
\quad \quad \quad \text{by } (\text{simp add: finite-bind}) \\
\quad \quad \text{ultimately have finite } ?expr \land \text{card } ?expr = 1 \text{ by blast} \\
\}
\]
moreover have finite $?S$
using $\langle \text{finite } B \rangle \langle \text{finite } P \rangle$ 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 $?S = j \sim (n - k)$
proof
\[
\quad \text{have card } (B - B') = n - k \\
\quad \quad \text{using } B' \langle \text{finite } B' \rangle \langle \text{card } B = n \rangle \\
\quad \quad \text{by } (\text{subst card-Diff-subset}) \text{ auto} \\
\quad \text{from this show } ?\text{thesis} \\
\quad \quad \text{using } \langle \text{finite } B \rangle \ P \\
\quad \quad \text{by } (\text{subst card-PiE}) (\text{simp add: prod-constant})+ \\
\quad \text{qed}
\]
ultimately have card $?expr = j \sim (n - k)$
by (simp add: card-bind-constant)
moreover have finite $?expr$
using $\langle \text{finite } ?S \rangle \langle \text{finite } \{P. \text{partition-on } C P \land \text{card } P = j\}\rangle$
by (auto intro!: finite-bind)
ultimately have finite $?expr \land \text{card } ?expr = j \sim (n - k)$ by blast
\} note inner = this
moreover have card $?S = \text{Bell } k$
using $B' \langle \text{finite } B' \rangle$ by (auto simp add: Bell-altdef[symmetric])
moreover have disjoint-family-on $?comp \ ?S$
using $P B'$
by (injectivity-solver rule: local.injectivity(2))
ultimately have card $?expr = j \sim (n - k) \ast \text{Bell } k$
by (subst card-bind-constant) auto
moreover have finite $?expr$
using inner $\langle \text{finite } ?S \rangle$ by (auto intro: finite-bind)
ultimately have finite $?expr \land \text{card } ?expr = j \sim (n - k) \ast \text{Bell } k$ by blast
\} note inner = this
moreover have card $?S = n \choose k$
using \langle \text{card } B = n \rangle \langle \text{finite } B \rangle \text{ by (simp add: n-subsets)}

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

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

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

ultimately have finite ?expr ∧ card ?expr = j ^ (n - k) * \binom{n}{k} * \text{Bell } k
  by blast

} note inner = this

moreover have finite ?S by simp

moreover have disjoint-family-on \text{?comp } ?S
  by (inj-solver rule: local.inj(5))

ultimately have card ?expr = \sum_{j \leq m} j ^ (n - k) * \binom{n}{k} * \text{Bell } k (\text{is - } ?\text{formula})
  using \langle \text{card } C = m \rangle \text{ by (subst card-bind)} (\text{auto intro: sum.cong})

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

ultimately have finite ?expr ∧ card ?expr = ?\text{formula} by blast

} note inner = this

moreover have finite ?S by simp

moreover have disjoint-family-on \text{?comp } ?S
  by (inj-solver rule: local.inj(6))

ultimately have step3: card (\text{construct-partition-on } B \text{ C}) = \sum_{k \leq n. \sum j \leq m. j} j ^ (n - k) * \binom{n}{k} * \text{Bell } k

unfolding \text{construct-partition-on-def}
  using \langle \text{card } B = n \rangle \text{ by (subst card-bind)} (\text{auto intro: sum.cong})

from step1 step2 step3 show \text{?thesis} by auto

qed

1.6 Corollaries of the Generalized Bell Recurrence

corollary \text{Bell-Stirling-eq:}
  \text{Bell } m = \sum_{j \leq m. \text{Stirling } m \text{ j}}

proof –
have \( \text{Bell } m = \text{Bell } (0 + m) \) by simp

also have \( \ldots = (\sum_{j \leq m} \text{Stirling } m j) \)

unfolding \( \text{Bell-eq[of 0]} \) by (simp add: Bell-0)

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

qed

corollary \( \text{Bell-recursive-eq} \):
\[
\text{Bell } (n + 1) = (\sum_{k \leq n} \text{n choose } k \times \text{Bell } k)
\]

unfolding \( \text{Bell-eq[of - 1]} \) by simp

end

References
