Spivey’s Generalized Recurrence for Bell Numbers

Lukas Bulwahn

April 19, 2020

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
1.1 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

lemma extensional-funcset-ext:
  assumes \( f \in A \to E \) \( g \in A \to E \)
  assumes \( \forall x. x \in A \implies f x = g x \)
  shows \( f = g \)
  using assms by (metis PiE-iff extensionalityI)

1.1.2 Additions for Injectivity Proofs

lemma inj-on-impl-inj-on-image:
  assumes inj-on \( f \) \( A \)
  assumes \( \forall x. x \in X \implies x \subseteq A \)
  shows inj-on \((\lambda x. x)\) \( X \)
  using assms by (meson inj-onI inj-on-image-eq-iff)

lemma injectivity-union:
  assumes \( A \cup B = C \cup D \)
  assumes \( P A P C \)
  assumes \( Q B Q D \)
  \( \forall S T. P S \implies Q T \implies S \cap T = \{\} \)
  shows \( A = C \land B = D \)
  using assms Int-Un-distrib Int-commute inf-sup-absorb by blast+

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

lemma injectivity-image-union:
  assumes \( (\lambda X. X \cup F X) \ ' P = (\lambda X. X \cup G X) \ ' P' \)
  assumes \( \forall X \in P. X \subseteq A \land X \in P'. X \subseteq A \)
  assumes \( \forall X \in P. \forall y \in F X. y \notin A \land \forall X \in P'. \forall y \in G X. y \notin A \)
  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) \( X \in P \) have \( X: (X \cup F X) \cap A = X \) 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

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 \( X = (X \cup F X) \cap A = X' \) by auto
moreover from assms(3,5) \( X' \in P' \) have \( X' = (X' \cup G X') \cap A = X' \) 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
\( \text{Bell} \ n = \text{card} \{ P. \ \text{partition-on} \{ 0 \ldots < n \} P \} \)

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: bijbetw \( \{ 0 \ldots < \text{card} \ A \} \ A \)
using ex-bijbetw-nat-finite by blast
from this have inj: injon \( \{ 0 \ldots < \text{card} \ A \} \)
using bijbetw-imp-inj-on by blast
from bij have image-f-eq: \( A = f' \{ 0 \ldots < \text{card} \ A \} \)
using bijbetw-imp-surj-on by blast
have \( \forall x \in \{ P. \ \text{partition-on} \{ 0 \ldots < \text{card} \ A \} \ P \}. \ x \subseteq \text{Pow} \{ 0 \ldots < \text{card} \ A \} \)
by (auto elim: partition-onE)
from this inj have injon ((\') ((\') f)) \{ P. \ \text{partition-on} \ \{ 0 \ldots < \text{card} \ A \} \ P \}
by (intro injon-impl-injon-image[of - Pow \{ 0 \ldots < \text{card} \ A \}])
injon-impl-injon-image[of - \{ 0 \ldots < \text{card} \ A \}] \ +
moreover from inj have ((\') ((\') f) \ P. \ \text{partition-on} \ \{ 0 \ldots < \text{card} \ A \} \ P \) = \{ P. \ partition-on \ A \ P \}
by (subst image-f-eq, auto elim!: set-of-partition-on-map)
ultimately have bijbetw ((\') ((\') f)) \{ P. \ \text{partition-on} \ \{ 0 \ldots < \text{card} \ A \} \ P \} \ P.
partition-on \ A \ P \)
by (auto intro: bijbetw-imageI)
from this \( \text{finite} \ A \) show \( \text{thesis} \)
unfolding Bell-def
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
fix $Q'$
assume $Q': Q' \in \{ Q', \text{partition-on (B} \cup \text{C)} \ Q' \}$
from $Q'$ have $\{ \} \notin Q'$ by (auto elim!: partition-onE)
obtain $Q$ where $Q: Q = ((\lambda X. \text{if } X \subseteq B \text{ then } X \text{ else } \{ \} \ ' Q') - \{ \} \}$ by blast
obtain $P'$ where $P': P' = ((\lambda X. \text{if } X \subseteq B \text{ then } \{ \} \ ' \text{else } X \ ' Q') - \{ \} \}$ 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 $\langle B \cap C = \{ \} \rangle$ 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
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 $X: x \in X \in Q'$
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 $X (x \in B - B' \cup C)$ have $\neg X \subseteq B$
using $\langle B \cap C = \{ \} \rangle$ by auto
from this $\langle x \in Q \rangle$ 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 $P: P = (\lambda X. X \cap C) ' P'$ 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 $f: f = (\lambda x. \text{if } x \in B - B' \text{ then } \text{THE } X. x \in X \land X \in P') \cap \ C \text{ else undefined} \}$ by blast
have $P'$-prop: $P' = (\lambda X. X \cup \{ x \in B - B', f x = X \})' P$
proof
\{ 
fix X 
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 $\langle x \in X \rangle$ show ?thesis by simp
  next
    assume $x \in B - B'$
    from partition-on-P' $\langle x \in X \rangle \langle x \in P' \rangle$ have (THE $X$. $x \in X \land X \in P'$)
      by (simp add: partition-on-the-part-eq)
    from $\langle x \in B - B' \rangle$ this show ?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 $x-in$: $x \in \{x \in B - B', f x = X \cap C\}$
    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 $X$. $x \in X \land X \in P'$) $\land C = X \cap C$
      unfolding $f$ by auto
    from P'-nosubset $\forall X \in P'$ 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 $y-in$: $y \in \{THE X. x \in X \land X \in P' \} \land C$ by simp
    from $y$ y-in have $y \in X y \in \{THE X. x \in X \land X \in P' \}$ 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 ext1 by (rule the1I2) auto
    ultimately have (THE $X$. $x \in X \land X \in P'$) $\in X$ using $\langle X \in P' \rangle$
      auto
    from this ext1 show ?thesis by (auto intro: the1I2)
qed
qed
from \langle X \in P', this have X \in (\lambda X. X \cup \{x \in B - B', f x = X\}) \cdot P \rangle unfolding P by simp
)
from this show P' \subseteq (\lambda X. X \cup \{x \in B - B', f x = X\}) \cdot P ..
next

\{ fix x assume x-in-image: x \in (\lambda X. X \cup \{x \in B - B', f x = X\}) \cdot P \}
\{ fix X assume X \in P' 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 in: \exists!X. x \in X \land X \in P'
  using partition-on-P' by (auto intro: partition-on-partition-on-unique)
from this have in-p: (THE X. x \in X \land X \in P') \in P'
  and x-in: x \in (THE X. x \in X \land X \in P')
  by (metis (mono-tags, lifting) theI)+
have f x = X \cap C \iff (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 (THE X. x \in X \land X \in P') = X
proof
  assume (THE X. x \in X \land X \in P') = X
from this show (THE X. x \in X \land X \in P') \cap C = X \cap C by auto
next
  assume (THE X. x \in X \land X \in P') \cap C = X \cap C
  have (THE X. x \in X \land X \in P') \cap X \neq \{}
    using P'-witness (THE X. x \in X \land X \in P') \cap C = X \cap C \iff X \in P' by fastforce
  from this show (THE X. x \in X \land X \in P') = X
    using partition-on-P'\cdot[unfolded partition-on-def disjoint-def] in-p \langle X \in P' by metis
qed
also have ... \iff x \in X
  using ex1 :X \in P' \cdot x-in by (auto; metis (no-types, lifting) the-equality)
finally have f x = X \cap C \iff x \in X .
\}
from this show ?thesis by auto
qed
moreover have X \subseteq B - B' \cup C
  using partition-on-P' \cdot X \in P' 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
\}

7
from this x-in-image have \( x \in P' \) unfolding \( P \) by auto
}

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

c qed

from partition-on-P' have \( f\)-prop: \( f \in (B - B') \rightarrow_E P \)

unfolding \( f \) by (auto simp add: partition-on-the-part-mem)

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

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

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

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

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

by (simp add: \( \text{assms}(2) \) partition-on-le-set-elements)

from \( \text{partition-on} \( C \) \( P' \) \( j \) have \( P\)-prop: partition-on \( C \) \( P \) \& card \( P = j \) by auto

from \( k \) \( (B' \subseteq B) \) have \( B'\)-prop: \( B' \subseteq B \) \& card \( B' = k \) by auto

show \( Q' \in \text{construct-partition-on} B C \)

using \( j\)-prop \( k\)-prop \( P'\)-prop \( B'\)-prop \( Q\)-prop \( P\)-prop \( Q'\)-prop

unfolding construct-partition-on-def

by (auto simp del: atLeastAtMost-iff) blast

c qed

1.4 Injectivity of the Set Construction

lemma injectivity:

\[ \text{assumes } B \cap C = \{\} \]

\[ \text{assumes } P: (\text{partition-on} \( C \) \( P \) \& card \( P = j \)) \& (\text{partition-on} \( C \) \( P' \) \& card \( P' = j' \)) \]

\[ \text{assumes } B': (B' \subseteq B \& card \( B' = k \)) \& (B'' \subseteq B \& card \( B'' = k' \)) \]

\[ \text{assumes } Q: \text{partition-on} B' Q \& \text{partition-on} B'' Q' \]

\[ \text{assumes } f: f \in B - B' \rightarrow_E P \& g \in B - B'' \rightarrow_E P' \]

\[ \text{assumes } P'= P'' = (\lambda X . X \cup \{x \in B - B'. f x = X\}) \& P \]

\[ \text{assumes } P''' = (\lambda X . X \cup \{x \in B - B'' . g x = X\}) \& P' \]

\[ \text{assumes } eq\text{-result}: P'' \cup Q = P''' \cup Q' \]

\[ \text{shows } f = g \text{ and } Q = Q' \text{ and } B' = B'' \]

and \( P = P' \) and \( j = j' \) and \( k = k' \)

proof –

\[ \text{have } P\text{-nonempty-sets: } \forall X \in P. \exists c \in C. c \in X \forall X \in P'. \exists c \in C. c \in X \]

using \( P \) by (force elim: partition-onE)+

\[ \text{have } 1: \forall X \in P''. \exists c \in C. c \in X \forall X \in P''' . \exists c \in C. c \in X \]

using \( P' \) P-nonempty-sets by fastforce+

\[ \text{have } 2: \forall X \in Q. \forall x \in X. x \notin C \forall X \in Q'. \forall x \in X. x \notin C \]

using \( B \cap C = \{\} \) \( Q B' \) by (auto elim: partition-onE)

from \( eq\text{-result} \) 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-eq-implies-eq-carrier)

have subset-C: \( \forall X \in P . X \subseteq C \forall X \in P' . X \subseteq C \)

using \( P \) by (auto elim: partition-onE)

have eq-image: \( (\lambda X . X \cup \{x \in B - B'. f x = X\}) \& P = (\lambda X . X \cup \{x \in B - \)
\[ B'' \setminus g \times X \} = P' \]

```
using \( P' \setminus P'' = P'' \setminus \) by auto
from this \( \{ B \cap C = \{ \} \) show \( P = P' \)
by (auto dest: injectivity-union[of \( \cdot \) subset-C])
have eq2: \((\lambda X. X \cup \{ x \in B - B', f x \times X \}) \cdot P = (\lambda X. X \cup \{ x \not\in B - B', g x = X \}) \cdot P \)
using \( P = P' \setminus B' = B'' \) eq-image by simp
from \( P \) have \( \text{ P-props: } \forall X \in P. \ X \subseteq C \implies \forall X \in P. \ X \not\in \{ \} \) by (auto elim: partition-onE)
have invert: \((\forall X \in P. \ (X \cup \{ x \in B - B', f x \times X \}) \cap C = X \cap (X \cup \{ x \in B - B', b x = X \}) \cap C = X) \)
using \( (B \cap C = \{ \}) \cdot \text{ P-props by auto} \)
have eq3: \((\forall X \in P. \ (X \cup \{ x \in B - B', f x = X \}) \cap C = X \cup \{ x \in B - B', g x = X \}) \)
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
taxfix \( X \)
assume \( X \in P \)
from this \( \text{ P have } X \subseteq C \) by (auto elim: partition-onE)
have disjoint: \( X \cap \{ x \in B - B', f x = X \} = \{ \} \ X \cap \{ x \in B - B', g x = X \} = \{ \} \)
using \( (B \cap C = \{ \}) \cdot X \subseteq C \) by auto
from eq2 \((X \in P) \) have \( X \cap \{ x \in B - B', f x = X \} = X \cap \{ x \in B - B', g x = X \} \)
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 \( f \) have eq6: \((\forall b \in B - B', f b = g b) \) by (auto intro: extensional-funcset-ext)
from \( P \) show \( \text{ P by simp} \)
from \( B' \cdot B' = B'' \) show \( k = k' \) by simp
```

```
qed
```

```

1.5 The Generalized Bell Recurrence Relation

```
by (simp add: construct-partition-on)
note injectivity = injectivity[OF `B ∩ C = `{}`]
let ?expr = 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}
}
{
  fix k
  assume k: k ∈ `{..card B}`
  let ?expr = ?comp k

  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 ∧ card P = j}`
    from this have partition-on C P by simp
    let ?expr = ?comp P
    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 ∧ card B' = k}`
      from this have B' ⊆ B by simp
      let ?expr = ?comp B'
      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 card (B − B’) = n − k
    using B’ ⟨finite B’⟩ ⟨card B = n⟩
    by (subt card-Diff-subset) auto
  from this show ?thesis
    using ⟨finite B⟩ ⟨finite P⟩
    by (subt card-PiE) (simp add: prod-constant)+
  qed
ultimately have finite ?expr
  by (simp add: card-bind-constant)
moreover have finite ?expr
  using ⟨finite ?S⟩ ⟨finite {P. partition-on C P ∧ card P = j}⟩
  by (auto intro!: finite-bind)
ultimately have finite ?expr ∧ card ?expr = j ¯ (n − k) by blast
}

moreover have card ?S = Bell k
  using B’ ⟨finite B’⟩ 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 ¯ (n − k) * Bell k
  by (subt card-bind-constant) auto
moreover have finite ?expr
  using inner ⟨finite ?S⟩
  by (auto intro: finite-bind)
ultimately have finite ?expr ∧ card ?expr = j ¯ (n − k) * Bell k by blast
}

note inner = this
moreover have card ?S = n choose k

11
1.6 Corollaries of the Generalized Bell Recurrence

corollary Bell-Stirling-eq:
 Bell m = (∑ j≤m. Stirling m j)
have $\text{Bell } m = \text{Bell } (0 + m)$ by simp
also have $... = (\sum_{j \leq m}. \text{Stirling } m j)$
unfolding $\text{Bell-eq[of 0]}$ by (simp add: Bell-0)
finally show $\varepsilon\text{thesis }$.
qed

corollary $\text{Bell-recursive-eq:}$
$\text{Bell } (n + 1) = (\sum_{k \leq n}. (n \text{ choose } k) \ast \text{Bell } k)$
unfolding $\text{Bell-eq[of - 1]}$ by simp
end

References
