Spivey's Generalized Recurrence for Bell Numbers

Lukas Bulwahn

March 17, 2025

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

theory *Bell-Numbers*
imports
HOL−Library.FuncSet

HOL−Library.Monad-Syntax
HOL−Library.Code-Target-Nat
HOL−Combinatorics.Stirling
Card-Partitions.Injectivity-Solver
Card-Partitions.Card-Partitions
begin

1.1 Preliminaries

1.1.1 Additions to FuncSet

lemma *extensional-funcset-ext*:
assumes $f \in A \rightarrow_E B$ $g \in A \rightarrow_E B$
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x = g\ x$
shows $f = g$
⟨*proof*⟩

1.1.2 Additions for Injectivity Proofs

lemma *inj-on-impl-inj-on-image*:
assumes *inj-on f A*
assumes $\bigwedge x.\ x \in X \Longrightarrow x \subseteq A$
shows *inj-on* $((`)\ f)\ X$
⟨*proof*⟩

lemma *injectivity-union*:
assumes $A \cup B = C \cup D$
assumes $P\ A$ $P\ C$
assumes $Q\ B$ $Q\ D$
$\bigwedge S\ T.\ P\ S \Longrightarrow Q\ T \Longrightarrow S \cap T = \{\}$
shows $A = C \wedge B = D$
⟨*proof*⟩

lemma *injectivity-image*:
assumes $f\ `\ A = g\ `\ A$
assumes $\forall x{\in}A.\ invert\ (f\ x) = x \wedge invert\ (g\ x) = x$
shows $\forall x{\in}A.\ f\ x = g\ x$
⟨*proof*⟩

lemma *injectivity-image-union*:
assumes $(\lambda X.\ X \cup F\ X)\ `\ P = (\lambda X.\ X \cup G\ X)\ `\ P'$
assumes $\forall X \in P.\ X \subseteq A$ $\forall X \in P'.\ X \subseteq A$
assumes $\forall X \in P.\ \forall y{\in}F\ X.\ y \notin A$ $\forall X \in P'.\ \forall y{\in}G\ X.\ y \notin A$
shows $P = P'$
⟨*proof*⟩

1.2 Definition of Bell Numbers

definition *Bell* :: *nat* ⇒ *nat*
where

$\mathit{Bell}\ n = \mathit{card}\ \{P.\ \mathit{partition\text{-}on}\ \{0..{<}n\}\ P\}$

lemma *Bell-altdef*:
assumes *finite A*
shows $\mathit{Bell}\ (\mathit{card}\ A) = \mathit{card}\ \{P.\ \mathit{partition\text{-}on}\ A\ P\}$
⟨*proof*⟩

lemma *Bell-0*:
$\mathit{Bell}\ 0 = 1$
⟨*proof*⟩

1.3 Construction of the Partitions

definition $\mathit{construct\text{-}partition\text{-}on} :: {'a}\ set \Rightarrow {'a}\ set \Rightarrow {'a}\ set\ set\ set$
where
$\mathit{construct\text{-}partition\text{-}on}\ B\ C =$
do {
$k \leftarrow \{0..\mathit{card}\ B\}$;
$j \leftarrow \{0..\mathit{card}\ C\}$;
$P \leftarrow \{P.\ \mathit{partition\text{-}on}\ C\ P \wedge \mathit{card}\ P = j\}$;
$B' \leftarrow \{B'.\ B' \subseteq B \wedge \mathit{card}\ B' = k\}$;
$Q \leftarrow \{Q.\ \mathit{partition\text{-}on}\ B'\ Q\}$;
$f \leftarrow (B - B') \rightarrow_E P$;
$P' \leftarrow \{(\lambda X.\ X \cup \{x \in B - B'.\ f\ x = X\})\ \text{‘}\ P\}$;
$\{P' \cup Q\}$
}

lemma *construct-partition-on*:
assumes *finite B finite C*
assumes $B \cap C = \{\}$
shows $\mathit{construct\text{-}partition\text{-}on}\ B\ C = \{P.\ \mathit{partition\text{-}on}\ (B \cup C)\ P\}$
⟨*proof*⟩

1.4 Injectivity of the Set Construction

lemma *injectivity*:
assumes $B \cap C = \{\}$
assumes P: $(\mathit{partition\text{-}on}\ C\ P \wedge \mathit{card}\ P = j) \wedge (\mathit{partition\text{-}on}\ C\ P' \wedge \mathit{card}\ P' = j')$
assumes B': $(B' \subseteq B \wedge \mathit{card}\ B' = k) \wedge (B'' \subseteq B \wedge \mathit{card}\ B'' = k')$
assumes Q: $\mathit{partition\text{-}on}\ B'\ Q \wedge \mathit{partition\text{-}on}\ B''\ Q'$
assumes f: $f \in B - B' \rightarrow_E P \wedge g \in B - B'' \rightarrow_E P'$
assumes P': $P'' = (\lambda X.\ X \cup \{x \in B - B'.\ f\ x = X\})\ \text{‘}\ P \wedge$
$P''' = (\lambda X.\ X \cup \{x \in B - B''.\ g\ x = X\})\ \text{‘}\ P'$
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*⟩

1.5 The Generalized Bell Recurrence Relation

theorem *Bell-eq*:
$Bell\ (n + m) = (\sum k \le n.\ \sum j \le m.\ j\ \hat{}\ (n - k) * Stirling\ m\ j * (n\ choose\ k) * Bell\ k)$
$\langle proof \rangle$

1.6 Corollaries of the Generalized Bell Recurrence

corollary *Bell-Stirling-eq*:
$Bell\ m = (\sum j \le m.\ Stirling\ m\ j)$
$\langle proof \rangle$

corollary *Bell-recursive-eq*:
$Bell\ (n + 1) = (\sum k \le n.\ (n\ choose\ k) * Bell\ k)$
$\langle proof \rangle$

1.7 Code equations for the computation of Bell numbers

It is slow to compute Bell numbers without dynamic programming (DP). The following is a DP algorithm derived from the previous recursion formula *Bell-recursive-eq*.

```
fun Bell-list-aux :: nat ⇒ nat list
  where
  Bell-list-aux 0 = [1] |
  Bell-list-aux (Suc n) = (
    let prev-list = Bell-list-aux n;
        next-val = (∑(k,z) ← List.enumerate 0 prev-list. z * (n choose (n−k)))
    in next-val#prev-list)

definition Bell-list :: nat ⇒ nat list
  where Bell-list n = rev (Bell-list-aux n)

lemma bell-list-eq: Bell-list n = map Bell [0..<n+1]
⟨proof⟩

lemma Bell-eval[code]: Bell n = last (Bell-list n)
  ⟨proof⟩

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 Integer Sequences*. https://oeis.org/A000110.

[2] M. Z. Spivey. A generalized recurrence for Bell numbers. *Journal of Integer Sequences*, 11, 2008. Electronic copy available at https://cs.uwaterloo.ca/journals/JIS/VOL11/Spivey/spivey25.pdf.