Enumeration of Equivalence Relations

Emin Karayel

March 17, 2025

Abstract

This entry contains a formalization of an algorithm enumerating all equivalence relations on an initial segment of the natural numbers. The approach follows the method described by Stanton and White [5, §1.5] using restricted growth functions.

The algorithm internally enumerates restricted growth functions (as lists), whose equivalence kernels then form the equivalence relations. This has the advantage that the representation is compact and lookup of the relation reduces to a list lookup operation.

The algorithm can also be used within a proof and an example application is included, where a sequence of variables is split by the possible partitions they can form.

1 Introduction

theory *Equivalence-Relation-Enumeration*
  imports *HOL−Library.Sublist HOL−Library.Disjoint-Sets*
    Card-Equiv-Relations.Card-Equiv-Relations
begin

As mentioned in the abstract the enumeration algorithm relies on the bijection between restricted growth functions (RGFs) of length n and the equivalence relations on $\{..<n\}$, where the bijection is the operation that forms the equivalence kernels of an RGF. The method is being dicussed, for example, by [3, 4] or [5, §1.5].

An enumeration algorithm for RGFs is less convoluted than one for equivalence relations or partitions and the representation has the advantage that checking whether a pair of elements are equivalent can be done by performing two list lookup operations.

After a few preliminary results in the following section, Section 3 introduces the enumeration algorithm for RGFs and shows that the function enumerates all of them (for the given length) without repetition. Section 4 shows that the operation of forming the equivalence kernel is a bijection and concludes with the correctness of the entire algorithm. In Section 5 an interesting application is being discussed, where the enumeration of partitions is applied within a proof. Section 6 contains a few additional results, such as the fact that the length of the enumerated list is a Bell number. The latter result relies on the formalization of the cardinality of equivalence relations by Bulwahn [2].

2 Preliminary Results

This section contains a few preliminary results used in the proofs below.

lemma *length-filter*: $length\ (filter\ p\ xs) = sum\text{-}list\ (map\ (\lambda x.\ of\text{-}bool\ (\ p\ x))\ xs)$
⟨*proof*⟩

lemma *count-list-expand*: $count\text{-}list\ xs\ x = length\ (filter\ ((=)\ x)\ xs)$
⟨*proof*⟩

An induction schema (similar to *list-induct2* and *rev-induct*) for two lists of equal length, where induction step is shown appending elements at the end.

lemma *list-induct-2-rev*[*consumes 1*, *case-names Nil Cons*]:
assumes $length\ x = length\ y$
assumes $P\ []\ []$
assumes $\bigwedge x\ xs\ y\ ys.\ length\ xs = length\ ys \Longrightarrow P\ xs\ ys \Longrightarrow P\ (xs@[x])\ (ys@[y])$
shows $P\ x\ y$
⟨*proof*⟩

If all but one value of a sum is zero then it can be evaluated on the remaining point:

lemma *sum-collapse*:
fixes $f :: 'a \Rightarrow 'b::\{comm\text{-}monoid\text{-}add\}$
assumes $finite\ A$
assumes $z \in A$
assumes $\bigwedge y.\ y \in A \Longrightarrow y \neq z \Longrightarrow f\ y = 0$
shows $sum\ f\ A = f\ z$
⟨*proof*⟩

Number of occurrences of elements in lists is preserved under injective maps.

lemma *count-list-inj-map*:
assumes $inj\text{-}on\ f\ (set\ x)$
assumes $y \in set\ x$
shows $count\text{-}list\ (map\ f\ x)\ (f\ y) = count\text{-}list\ x\ y$
⟨*proof*⟩

A relation cannot be an equivalence relation on two distinct sets.

lemma *equiv-on-unique*:
assumes $equiv\ A\ p$
assumes $equiv\ B\ p$
shows $A = B$
⟨*proof*⟩

The restriction of an equivalence relation is itself an equivalence relation.

lemma *equiv-subset*:
  assumes $B \subseteq A$
  assumes *equiv* A p
  shows *equiv* B (*Restr* p B)
⟨*proof*⟩

3 Enumerating Restricted Growth Functions

fun *rgf-limit* :: *nat list* ⇒ *nat*
  where
    rgf-limit [] = *0* |
    rgf-limit (*x#xs*) = *max* (*x+1*) (*rgf-limit xs*)

lemma *rgf-limit-snoc*: *rgf-limit* (*x*@[*y*]) = *max* (*y+1*) (*rgf-limit x*)
  ⟨*proof*⟩

lemma *rgf-limit-ge*: $y \in$ *set xs* $\Longrightarrow$ $y <$ *rgf-limit xs*
  ⟨*proof*⟩

definition *rgf* :: *nat list* ⇒ *bool*
  where *rgf x* = (∀ *ys y*. *prefix* (*ys*@[*y*]) *x* ⟶ $y \leq$ *rgf-limit ys*)

The function *rgf-limit* returns the smallest natural number larger than all list elements, it is the largest allowed value following *xs* for restricted growth functions. The definition *rgf* is the predicate capturing the notion.

fun *enum-rgfs* :: *nat* ⇒ (*nat list*) *list*
  where
    enum-rgfs 0 = [[]] |
    enum-rgfs (*Suc n*) = [(*x*@[*y*]). *x* ← *enum-rgfs n*, *y* ← [*0*..<*rgf-limit x+1*]]

The function *enum-rgfs n* returns all RGFs of length *n* without repetition. The fact is verified in the three lemmas at the end of this section.

lemma *rgf-snoc*:
  rgf (*xs*@[*x*]) ⟷ *rgf xs* ∧ $x <$ *rgf-limit xs* + *1*
  ⟨*proof*⟩

lemma *rgf-imp-initial-segment*:
  rgf xs $\Longrightarrow$ *set xs* = {..<*rgf-limit xs*}
⟨*proof*⟩

lemma *enum-rgfs-returns-rgfs*:
  assumes $x \in$ *set* (*enum-rgfs n*)
  shows *rgf x*
  ⟨*proof*⟩

lemma *enum-rgfs-len*:
  assumes $x \in$ *set* (*enum-rgfs n*)

shows *length* $x = n$
⟨*proof*⟩

lemma *equiv-rels-enum*:
assumes *rgf* x
shows *count-list* (*enum-rgfs* (*length* x)) $x = 1$
⟨*proof*⟩

4 Enumerating Equivalence Relations

The following definition returns the equivalence relation induced by a list, for example, by a restricted growth function.

definition *kernel-of* :: $'a$ *list* $\Rightarrow$ *nat rel*
where *kernel-of xs* $= \{(i,j).\ i <$ *length xs* $\wedge\ j <$ *length xs* $\wedge$ *xs* ! $i =$ *xs* ! $j\}$

Using that the enumeration function for equivalence relations on $\{..<n\}$ is straight-forward to define:

definition *equiv-rels* **where** *equiv-rels* $n =$ *map kernel-of* (*enum-rgfs* n)

The following lemma shows that the image of *kernel-of* is indeed an equivalence relation:

lemma *kernel-of-equiv*: *equiv* $\{..<$*length xs*$\}$ (*kernel-of xs*)
⟨*proof*⟩

lemma *kernel-of-eq-len*:
assumes *kernel-of* $x =$ *kernel-of* y
shows *length* $x =$ *length* y
⟨*proof*⟩

lemma *kernel-of-eq*:
(*kernel-of* $x =$ *kernel-of* y) $\longleftrightarrow$
(*length* $x =$ *length* y $\wedge$ ($\forall j <$ *length* x. $\forall i < j$. (x ! $i = x$! j) $=$ (y ! $i = y$! j)))
⟨*proof*⟩

lemma *kernel-of-snoc*:
kernel-of (*xs*) $=$ *Restr* (*kernel-of* (*xs*@[x])) $\{..<$*length xs*$\}$
⟨*proof*⟩

lemma *kernel-of-inj-on-rgfs-aux*:
assumes *length* $x =$ *length* y
assumes *rgf* x
assumes *rgf* y
assumes *kernel-of* $x =$ *kernel-of* y
shows $x = y$
⟨*proof*⟩

lemma *kernel-of-inj-on-rgfs*:

inj-on kernel-of $\{x.\ rgf\ x\}$
⟨*proof*⟩

Applying an injective map to a list preserves the induced relation:

lemma *kernel-of-under-inj-map*:
assumes *inj-on f* (*set x*)
shows *kernel-of x* = *kernel-of* (*map f x*)
⟨*proof*⟩

lemma *all-rels-are-kernels*:
assumes *equiv* $\{..<n\}$ *p*
shows $\exists(x :: nat\ set\ list).\ kernel\text{-}of\ x = p \wedge length\ x = n$
⟨*proof*⟩

For any list there is always an injective map on its set, such that its image is an RGF.

lemma *map-list-to-rgf*:
$\exists f.\ inj\text{-}on\ f\ (set\ x) \wedge rgf\ (map\ f\ x)$
⟨*proof*⟩

For any relation there is a corresponding RGF:

lemma *rgf-exists*:
assumes *equiv* $\{..<n\}$ *r*
shows $\exists x.\ rgf\ x \wedge length\ x = n \wedge kernel\text{-}of\ x = r$
⟨*proof*⟩

These are the main result of this entry: The function *equiv-rels n* enumerates the equivalence relations on $\{..<n\}$ without repetition.

theorem *equiv-rels-set*:
assumes $x \in set\ (equiv\text{-}rels\ n)$
shows *equiv* $\{..<n\}$ *x*
⟨*proof*⟩

theorem *equiv-rels*:
assumes *equiv* $\{..<n\}$ *r*
shows *count-list* (*equiv-rels n*) *r* = *1*
⟨*proof*⟩

A corollary of the previous theorem is that the sum of the indicator function for a relation over *equiv-rels n* is always one.

corollary *equiv-rels-2*:
assumes $n = length\ xs$
shows $(\sum x{\leftarrow}equiv\text{-}rels\ n.\ of\text{-}bool\ (kernel\text{-}of\ xs = x)) = (1 :: 'a :: \{semiring\text{-}1\})$
⟨*proof*⟩

5 Example Application

In this section, I wanted to discuss an interesting application within the context of a proof in Isabelle. This is motivated by a real-world example [1, §2.2], where a function in a 4-times iterated sum could only be reduced by splitting it according to the equivalence relation formed by the indices. The notepad below illustrates how this can be done (in the case of 3 index variables).

notepad
begin
  fix f :: *nat* $\times$ *nat* $\times$ *nat* $\Rightarrow$ *nat*
  fix I :: *nat set*
  assume *a:finite I*

To be able to break down such a sum by partitions let us introduce the function P which is defined to be sum of an indicator function over all possible equivalence relations its argument can form:

  define P :: *nat list* $\Rightarrow$ *nat*
    where $P = (\lambda xs.\ (\sum x \leftarrow \textit{equiv-rels}\ (\textit{length}\ xs).\ \textit{of-bool}\ (\textit{kernel-of}\ xs = x)\ ))$

Note that its value is always one, hence we can introduce it in an algebraic equation easily:

  have *P-one*: $\bigwedge xs.\ P\ xs = 1$
    by (*simp add*: *P-def equiv-rels-2*)

  note *unfold-equiv-rels* = *P-def equiv-rels-def numeral-eq-Suc kernel-of-eq*
    neq-commute All-less-Suc comp-def

  define r **where** $r = (\sum i \in I.\ (\sum j \in I.\ (\sum k \in I.\ \ f\ (i,j,k))))$

As a first step, we just introduce the factor $P\ [i,\ j,\ k]$.

  have $r = (\sum i \in I.\ (\sum j \in I.\ (\sum k \in I.\ \ f\ (i,j,k) * P\ [i,j,k])))$
    by (*simp add*:*P-one r-def cong*:*sum.cong*)

By expanding the definition of P and distributing, the sum can be expanded into 5 sums each representing a distinct equivalence relation formed by the indices.

  also have ... =
    $(\sum i{\in}I.\ f\ (i,\ i,\ i))\ +$
    $(\sum i{\in}I.\ \sum j{\in}I.\ f\ (i,\ i,\ j) * \textit{of-bool}\ (i \neq j))\ +$
    $(\sum i{\in}I.\ \sum j{\in}I.\ f\ (i,\ j,\ i) * \textit{of-bool}\ (i \neq j))\ +$
    $(\sum i{\in}I.\ \sum j{\in}I.\ f\ (i,\ j,\ j) * \textit{of-bool}\ (i \neq j))\ +$
    $(\sum i{\in}I.\ \sum j{\in}I.\ \sum k{\in}I.\ f\ (i,\ j,\ k) * \textit{of-bool}\ (j \neq k \wedge i \neq k \wedge i \neq j))$
    (**is** - = *?rhs*)
    by (*simp add*:*unfold-equiv-rels sum.distrib distrib-left sum-collapse*[*OF a*])
  finally have r = *?rhs* **by** *simp*
end

6 Additional Results

If two lists induce the same equivalence relation, then there is a bijection between the sets that preserves the multiplicities of its elements.

lemma *kernel-of-eq-imp-bij*:
  assumes $\textit{kernel-of}\ x = \textit{kernel-of}\ y$
  shows $\exists f.\ \textit{bij-betw}\ f\ (\textit{set}\ x)\ (\textit{set}\ y) \wedge$
    $(\forall z \in \textit{set}\ x.\ \textit{count-list}\ x\ z = \textit{count-list}\ y\ (f\ z))$
$\langle \textit{proof} \rangle$

As expected the length of *equiv-rels n* is the n-th Bell number.

lemma *len-equiv-rels*: $\textit{length}\ (\textit{equiv-rels}\ n) = \textit{Bell}\ n$
$\langle \textit{proof} \rangle$

Instead of forming an equivalence relation from a list, it is also possible to induce a partition from it:

definition *induced-par* $::$ $'a\ \textit{list} \Rightarrow \textit{nat set set}$ **where**
  $\textit{induced-par}\ xs = (\lambda k.\ \{i.\ i < \textit{length}\ xs \wedge xs\ !\ i = k\})\ \text{`}\ (\textit{set}\ xs)$

The following lemma verifies the commutative diagram, i.e., *induced-par xs* is the same partition as the quotient of $\{..<\textit{length}\ xs\}$ over the corresponding equivalence relation.

lemma *quotient-of-kernel-is-induced-par*:
  $\{..<\textit{length}\ xs\}\ //\ (\textit{kernel-of}\ xs) = (\textit{induced-par}\ xs)$
$\langle \textit{proof} \rangle$

end

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[2] L. Bulwahn. Cardinality of equivalence relations. *Archive of Formal Proofs*, May 2016. https://isa-afp.org/entries/Card_Equiv_Relations.html, Formal proof development.

[3] G. Hutchinson. Partioning algorithms for finite sets. *Commun. ACM*, 6(10):613–614, Oct. 1963.

[4] S. Milne. Restricted growth functions and incidence relations of the lattice of partitions of an n-set. *Advances in Mathematics*, 26(3):290–305, 1977.

[5] D. Stanton and D. White. *Constructive Combinatorics*. Springer, 1986.