# Enumeration of Equivalence Relations 

Emin Karayel

March 11, 2024


#### 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<br>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 $[5, \S 1.5]$.
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-list (map ( }\lambdax\mathrm{ x.of-bool ( p x) ) xs)
    <proof\rangle
lemma count-list-expand:count-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 \(x s=\) length \(y s \Longrightarrow P x s\) ys \(\Longrightarrow P(x s @[x])(y s @[y])\)
    shows \(P x y\)
    \(\langle p r o o f\rangle\)
```

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

```
lemma sum-collapse:
    fixes \(f::{ }^{\prime} a \Rightarrow{ }^{\prime} b::\{\) comm-monoid-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-on f (set x)
    assumes }y\in\mathrm{ set }
    shows count-list (map fx) (fy)= count-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=
    <proof>
```

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

```
lemma equiv-subset:
    assumes B\subseteqA
    assumes equiv A p
    shows equiv B (Restr p B)
<proof\rangle
```


## 3 Enumerating Restricted Growth Functions

fun rgf－limit $::$ nat list $\Rightarrow$ nat
where
rgf-limit []$=0 \mid$
rgf-limit $(x \# x s)=\max (x+1)($ rgf-limit $x s)$
lemma rgf-limit-snoc: rgf-limit $(x @[y])=\max (y+1)($ rgf-limit $x)$
〈proof〉
lemma rgf-limit-ge: $y \in$ set $x s \Longrightarrow y<r g f-l i m i t ~ x s ~$
$\langle p r o o f\rangle$
definition rgf :: nat list $\Rightarrow$ bool
where rgf $x=(\forall y s y$. prefix $(y s @[y]) x \longrightarrow y \leq r g f-l i m i t ~ y s)$

The function rgf－limit returns the smallest natural number larger than all list elements，it is the largest allowed value following $x s$ for restricted growth functions．The definition $r g f$ is the predicate capturing the notion．

```
fun enum-rgfs :: nat }=>\mathrm{ (nat list) list
    where
        enum-rgfs 0 = [[]] |
        enum-rgfs (Suc n) = [(x@[y]). x enum-rgfs n, y\leftarrow[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:
```

    \(r g f(x s @[x]) \longleftrightarrow r g f x s \wedge x<r g f-l i m i t ~ x s+1\)
    〈proof〉
    lemma rgf-imp-initial-segment:
rgf $x s \Longrightarrow$ set $x s=\{. .<$ rgf-limit $x s\}$
$\langle p r o o f\rangle$
lemma enum-rgfs-returns-rgfs:
assumes $x \in$ set (enum-rgfs $n$ )
shows rgf $x$
$\langle p r o o f\rangle$
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 $x s=\{(i, j) . i<$ length $x s \wedge j<$ length $x s \wedge x s!i=x s!j\}$
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 equiv－ alence relation：
lemma kernel－of－equiv：equiv $\{. .<$ length $x s\}$（kernel－of $x s$ ）
$\langle p r o o f\rangle$
lemma kernel－of－eq－len
assumes kernel－of $x=$ kernel－of $y$ shows length $x=$ length $y$
$\langle p r o o f\rangle$
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 $(x s)=\operatorname{Restr}($ kernel－of $(x s @[x])) \quad\{. .<$ length $x s\}$
＜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\rangle
```

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 \(f x\) )
\(\langle p r o o f\rangle\)
lemma all-rels-are-kernels:
    assumes equiv \(\{. .<n\} p\)
    shows \(\exists(x::\) nat set list \()\). kernel-of \(x=p \wedge\) length \(x=n\)
\(\langle p r o o f\rangle\)
```

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

```
lemma map-list-to-rgf:
    \existsf.inj-onf (set x)^rgf (map f x )
\langleproof\rangle
```

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-of $x=r$
$\langle p r o o f\rangle$
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 set (equiv-rels n)
    shows equiv {..<n} x
    \langleproof\rangle
```

```
theorem equiv-rels:
    assumes equiv {..<n} r
    shows count-list (equiv-rels n) r=1
<proof\rangle
```

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 (\sumx\leftarrowequiv-rels n. of-bool (kernel-of xs = x)) =(1 :: 'a :: {semiring-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 $[1, \S 2.2]$, 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\mathrm{ nat }\times\mathrm{ nat }=>\mathrm{ 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 \(P=\left(\lambda x s .\left(\sum x \leftarrow\right.\right.\) equiv-rels (length \(\left.x s\right)\). of-bool \((\) kernel-of \(\left.\left.x s=x)\right)\right)\)
```

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

```
have \(P\)-one: \(\bigwedge x s . P\) 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 \(r=\left(\sum i \in I .\left(\sum j \in I .\left(\sum k \in I . f(i, j, k)\right)\right)\right)\)
```

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

```
have r=(\sumi\inI. (\sumj\inI. (\sumk\inI. 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...\(=\)
    \(\left(\sum i \in I . f(i, i, i)\right)+\)
    \(\left(\sum i \in I . \sum j \in I . f(i, i, j) *\right.\) of-bool \(\left.(i \neq j)\right)+\)
    \(\left(\sum i \in I . \sum j \in I . f(i, j, i) *\right.\) of-bool \(\left.(i \neq j)\right)+\)
    \(\left(\sum i \in I . \sum j \in I . f(i, j, j) *\right.\) of-bool \(\left.(i \neq j)\right)+\)
    ( \(\sum i \in I . \sum j \in I . \sum k \in I . f(i, j, k) *\) of-bool \(\left.(j \neq k \wedge i \neq k \wedge i \neq j)\right)\)
    (is - = ? rhs )
    by (simp add:unfold-equiv-rels sum.distrib distrib-left sum-collapse[OF a])
finally have \(r=?\) ? \(h s\) 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 kernel-of x=kernel-of y
    shows \existsf. bij-betw f(set x) (set y)^
        (}\forallz\in\mathrm{ set x. count-list x z = count-list y (fz))
<proof\rangle
```

As expected the length of equiv-rels $n$ is the $n$-th Bell number.
lemma len-equiv-rels: length (equiv-rels $n$ ) Bell $n$〈proof〉

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

```
definition induced-par :: 'a list => nat set set where
    induced-par xs = ( }\lambdak.{i.i<length xs \wedge xs !i=k})'(set xs
```

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

```
lemma quotient-of-kernel-is-induced-par:
    {..<length xs } // (kernel-of xs) = (induced-par xs)
<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.

