

The Twelfold Way

Lukas Bulwahn

May 26, 2024

Abstract

This entry provides all cardinality theorems of the Twelfold Way. The Twelfold Way [1, 5, 6] systematically classifies twelve related combinatorial problems concerning two finite sets, which include counting permutations, combinations, multisets, set partitions and number partitions. This development builds upon the existing formal developments [2, 3, 4] with cardinality theorems for those structures. It provides twelve bijections from the various structures to different equivalence classes on finite functions, and hence, proves cardinality formulae for these equivalence classes on finite functions.

Contents

1 Preliminaries	4
1.1 Additions to Finite Set Theory	4
1.2 Additions to Equiv Relation Theory	4
1.2.1 Counting Sets by Splitting into Equivalence Classes	7
1.3 Additions to FuncSet Theory	8
1.4 Additions to Permutations Theory	9
1.5 Additions to List Theory	10
1.6 Additions to Disjoint Set Theory	12
1.7 Additions to Multiset Theory	13
1.8 Additions to Number Partitions Theory	15
1.9 Cardinality Theorems with Iverson Function	15
2 Main Observations on Operations and Permutations	16
2.1 Range Multiset	16
2.1.1 Existence of a Suitable Finite Function	16
2.1.2 Existence of Permutation	17
2.2 Domain Partition	19
2.2.1 Existence of a Suitable Finite Function	19
2.2.2 Equality under Permutation Application	21
2.2.3 Existence of Permutation	21
2.3 Number Partition of Range	26

2.3.1	Existence of a Suitable Finite Function	26
2.3.2	Equality under Permutation Application	27
2.3.3	Existence of Permutation	28
2.4	Bijections on Same Domain and Range	35
2.4.1	Existence of Domain Permutation	35
2.4.2	Existence of Range Permutation	36
3	Definition of Equivalence Classes	37
3.1	Permutation on the Domain	37
3.1.1	Respecting Functions	38
3.2	Permutation on the Range	40
3.2.1	Respecting Functions	41
3.3	Permutation on the Domain and the Range	43
3.3.1	Respecting Functions	45
4	Functions from A to B	49
4.1	Definition of Bijections	49
4.2	Properties for Bijections	49
4.3	Bijections	51
4.4	Cardinality	52
5	Injections from A to B	53
5.1	Properties for Bijections	53
5.2	Bijections	55
5.3	Cardinality	56
6	Functions from A to B, up to a Permutation of A	57
6.1	Definition of Bijections	57
6.2	Properties for Bijections	57
6.3	Bijections	60
6.4	Cardinality	61
7	Injections from A to B up to a Permutation of A	61
7.1	Definition of Bijections	61
7.2	Properties for Bijections	62
7.3	Bijections	65
7.4	Cardinality	67
8	Surjections from A to B up to a Permutation on A	68
8.1	Properties for Bijections	68
8.2	Bijections	69
8.3	Cardinality	70

9	Functions from A to B up to a Permutation on B	70
9.1	Definition of Bijections	70
9.2	Properties for Bijections	71
9.3	Bijections	74
9.4	Cardinality	75
10	Injections from A to B up to a Permutation on B	75
10.1	Properties for Bijections	75
10.2	Bijections	77
10.3	Cardinality	77
11	Surjections from A to B up to a Permutation on B	78
11.1	Properties for Bijections	78
11.2	Bijections	80
11.3	Cardinality	81
12	Surjections from A to B	81
13	Functions from A to B up to a Permutation on A and B	83
13.1	Definition of Bijections	84
13.2	Properties for Bijections	84
13.3	Bijections	88
13.4	Cardinality	89
14	Injections from A to B up to a permutation on A and B	89
14.1	Properties for Bijections	89
14.2	Bijections	91
14.3	Cardinality	93
15	Surjections from A to B up to a Permutation on A and B	93
15.1	Properties for Bijections	93
15.2	Bijections	95
15.3	Cardinality	96
16	Cardinality of Bijections	96
16.1	Bijections from A to B	97
16.2	Bijections from A to B up to a Permutation on A	97
16.3	Bijections from A to B up to a Permutation on B	98
16.4	Bijections from A to B up to a Permutation on A and B	99
17	Direct Proofs for Cardinality of Bijections	100
17.1	Bijections from A to B up to a Permutation on A	101
17.1.1	Equivalence Class	101
17.1.2	Cardinality	103
17.2	Bijections from A to B up to a Permutation on B	103

17.2.1	Equivalence Class	103
17.2.2	Cardinality	105
17.3	Bijections from A to B up to a Permutation on A and B . . .	105
17.3.1	Equivalence Class	105
17.3.2	Cardinality	107

18 The Twelfold Way 108

1 Preliminaries

```

theory Preliminaries
imports
  Main
  HOL-Library.Multiset
  HOL-Library.FuncSet
  HOL-Combinatorics.Permutations
  HOL-ex.Birthday-Paradox
  Card-Partitions.Card-Partitions
  Bell-Numbers-Spivey.Bell-Numbers
  Card-Multisets.Card-Multisets
  Card-Number-Partitions.Card-Number-Partitions
begin

```

1.1 Additions to Finite Set Theory

```

lemma subset-with-given-card-exists:
  assumes  $n \leq \text{card } A$ 
  shows  $\exists B \subseteq A. \text{card } B = n$ 
using assms proof (induct n)
  case 0
  then show ?case by auto
next
  case (Suc n)
  from this obtain B where  $B \subseteq A$   $\text{card } B = n$  by auto
  from this  $\langle B \subseteq A \rangle \langle \text{card } B = n \rangle$  have  $\text{card } B < \text{card } A$ 
    using Suc.prems by linarith
  from  $\langle \text{Suc } n \leq \text{card } A \rangle$  card.infinite have finite A by force
  from this  $\langle B \subseteq A \rangle$  finite-subset have finite B by blast
  from  $\langle \text{card } B < \text{card } A \rangle \langle B \subseteq A \rangle$  obtain a where  $a \in A$   $a \notin B$ 
    by (metis less-irrefl subsetI subset-antisym)
  have  $\text{insert } a B \subseteq A$   $\text{card } (\text{insert } a B) = \text{Suc } n$ 
    using  $\langle \text{finite } B \rangle \langle a \in A \rangle \langle a \notin B \rangle \langle B \subseteq A \rangle \langle \text{card } B = n \rangle$  by auto
  then show ?case by blast
qed

```

1.2 Additions to Equiv Relation Theory

```

lemmas univ-commute' = univ-commute[unfolded Equiv-Relations.proj-def]

```

lemma *univ-predicate-impl-forall*:

assumes *equiv A R*
assumes *P respects R*
assumes $X \in A // R$
assumes *univ P X*
shows $\forall x \in X. P x$

proof –

from *assms(1,3)* **obtain** x **where** $x \in X$
by (*metis equiv-class-self quotientE*)
from $\langle x \in X \rangle$ *assms(1,3)* **have** $X = R \text{ `` } \{x\}$
by (*metis Image-singleton-iff equiv-class-eq quotientE*)
from *assms(1,2,4)* **this show** *?thesis*
using *equiv-class-eq-iff univ-commute'* **by** *fastforce*

qed

lemma *univ-preserves-predicate*:

assumes *equiv A r*
assumes *P respects r*
shows $\{x \in A. P x\} // r = \{X \in A // r. \text{univ } P X\}$

proof

show $\{x \in A. P x\} // r \subseteq \{X \in A // r. \text{univ } P X\}$

proof

fix X

assume $X \in \{x \in A. P x\} // r$

from *this* **obtain** x **where** $x \in \{x \in A. P x\}$ **and** $X = r \text{ `` } \{x\}$

using *quotientE* **by** *blast*

have $X \in A // r$

using $\langle X = r \text{ `` } \{x\} \rangle$ $\langle x \in \{x \in A. P x\} \rangle$

by (*auto intro: quotientI*)

moreover have *univ P X*

using $\langle X = r \text{ `` } \{x\} \rangle$ $\langle x \in \{x \in A. P x\} \rangle$ *assms*

by (*simp add: proj-def[symmetric] univ-commute*)

ultimately show $X \in \{X \in A // r. \text{univ } P X\}$ **by** *auto*

qed

next

show $\{X \in A // r. \text{univ } P X\} \subseteq \{x \in A. P x\} // r$

proof

fix X

assume $X \in \{X \in A // r. \text{univ } P X\}$

from *this* **have** $X \in A // r$ **and** *univ P X* **by** *auto*

from $\langle X \in A // r \rangle$ **obtain** $x \in A$ **and** $X = r \text{ `` } \{x\}$

using *quotientE* **by** *blast*

have $x \in \{x \in A. P x\}$

using $\langle x \in A \rangle$ $\langle X = r \text{ `` } \{x\} \rangle$ $\langle \text{univ } P X \rangle$ *assms*

by (*simp add: proj-def[symmetric] univ-commute*)

from *this* **show** $X \in \{x \in A. P x\} // r$

using $\langle X = r \text{ `` } \{x\} \rangle$ **by** (*auto intro: quotientI*)

qed

qed

lemma *Union-quotient-restricted:*

assumes *equiv A r*

assumes *P respects r*

shows $\bigcup(\{x \in A. P x\} // r) = \{x \in A. P x\}$

proof

show $\bigcup(\{x \in A. P x\} // r) \subseteq \{x \in A. P x\}$

proof

fix *x*

assume $x \in \bigcup(\{x \in A. P x\} // r)$

from *this* obtain *X* where $x \in X$ and $X \in \{x \in A. P x\} // r$ by *blast*

from *this* obtain *x'* where $X = r \text{ `` } \{x'\}$ and $x' \in \{x \in A. P x\}$

using *quotientE* by *blast*

from *this* $\langle x \in X \rangle$ have $x \in A$

using $\langle \text{equiv } A \text{ } r \rangle$ by (*simp add: equiv-class-eq-iff*)

moreover from $\langle X = r \text{ `` } \{x'\} \rangle \langle x \in X \rangle \langle x' \in \{x \in A. P x\} \rangle$ have *P x*

using $\langle P \text{ respects } r \rangle$ *congruentD* by *fastforce*

ultimately show $x \in \{x \in A. P x\}$ by *auto*

qed

next

show $\{x \in A. P x\} \subseteq \bigcup(\{x \in A. P x\} // r)$

proof

fix *x*

assume $x \in \{x \in A. P x\}$

from *this* have $x \in r \text{ `` } \{x\}$

using $\langle \text{equiv } A \text{ } r \rangle$ *equiv-class-self* by *fastforce*

from $\langle x \in \{x \in A. P x\} \rangle$ have $r \text{ `` } \{x\} \in \{x \in A. P x\} // r$

by (*auto intro: quotientI*)

from *this* $\langle x \in r \text{ `` } \{x\} \rangle$ show $x \in \bigcup(\{x \in A. P x\} // r)$ by *auto*

qed

qed

lemma *finite-equiv-implies-finite-carrier:*

assumes *equiv A R*

assumes *finite (A // R)*

assumes $\forall X \in A // R. \text{finite } X$

shows *finite A*

proof –

from $\langle \text{equiv } A \text{ } R \rangle$ have $A = \bigcup(A // R)$

by (*simp add: Union-quotient*)

from *this* $\langle \text{finite } (A // R) \rangle \langle \forall X \in A // R. \text{finite } X \rangle$ show *finite A*

using *finite-Union* by *fastforce*

qed

lemma *finite-quotient-iff:*

assumes *equiv A R*

shows $\text{finite } A \iff (\text{finite } (A // R) \wedge (\forall X \in A // R. \text{finite } X))$

using *assms* by (*meson equiv-type finite-equiv-class finite-equiv-implies-finite-carrier*)

finite-quotient)

1.2.1 Counting Sets by Splitting into Equivalence Classes

lemma *card-equiv-class-restricted*:

assumes *finite* $\{x \in A. P x\}$

assumes *equiv* $A R$

assumes P respects R

shows $\text{card } \{x \in A. P x\} = \text{sum card } (\{x \in A. P x\} // R)$

proof –

have $\text{card } \{x \in A. P x\} = \text{card } (\bigcup (\{x \in A. P x\} // R))$

using $\langle \text{equiv } A R \rangle \langle P \text{ respects } R \rangle$ **by** (*simp add: Union-quotient-restricted*)

also have $\text{card } (\bigcup (\{x \in A. P x\} // R)) = (\sum C \in \{x \in A. P x\} // R. \text{card } C)$

proof –

from $\langle \text{finite } \{x \in A. P x\} \rangle$ **have** *finite* $(\{x \in A. P x\} // R)$

using $\langle \text{equiv } A R \rangle$ **by** (*metis finite-imageI proj-image*)

moreover from $\langle \text{finite } \{x \in A. P x\} \rangle$ **have** $\forall C \in \{x \in A. P x\} // R. \text{finite } C$

using $\langle \text{equiv } A R \rangle \langle P \text{ respects } R \rangle$ *Union-quotient-restricted*

Union-upper finite-subset by fastforce

moreover have $\forall C1 \in \{x \in A. P x\} // R. \forall C2 \in \{x \in A. P x\} // R. C1 \neq C2 \longrightarrow C1 \cap C2 = \{\}$

using $\langle \text{equiv } A R \rangle$ *quotient-disj*

by (*metis (no-types, lifting) mem-Collect-eq quotientE quotientI*)

ultimately show *?thesis*

by (*subst card-Union-disjoint*) (*auto simp: pairwise-def disjnt-def*)

qed

finally show *?thesis* .

qed

lemma *card-equiv-class-restricted-same-size*:

assumes *equiv* $A R$

assumes P respects R

assumes $\bigwedge F. F \in \{x \in A. P x\} // R \implies \text{card } F = k$

shows $\text{card } \{x \in A. P x\} = k * \text{card } (\{x \in A. P x\} // R)$

proof *cases*

assume *finite* $\{x \in A. P x\}$

have $\text{card } \{x \in A. P x\} = \text{sum card } (\{x \in A. P x\} // R)$

using $\langle \text{finite } \{x \in A. P x\} \rangle \langle \text{equiv } A R \rangle \langle P \text{ respects } R \rangle$

by (*simp add: card-equiv-class-restricted*)

also have $\text{sum card } (\{x \in A. P x\} // R) = k * \text{card } (\{x \in A. P x\} // R)$

by (*simp add: $\langle \bigwedge F. F \in \{x \in A. P x\} // R \implies \text{card } F = k \rangle$*)

finally show *?thesis* .

next

assume *infinite* $\{x \in A. P x\}$

from this have *infinite* $(\bigcup (\{a \in A. P a\} // R))$

using $\langle \text{equiv } A R \rangle \langle P \text{ respects } R \rangle$ **by** (*simp add: Union-quotient-restricted*)

from this have *infinite* $(\{x \in A. P x\} // R) \vee (\exists X \in \{x \in A. P x\} // R. \text{infinite } X)$

by *auto*

```

from this show ?thesis
proof
  assume infinite ( $\{x \in A. P\ x\} // R$ )
  from this  $\langle \textit{infinite } \{x \in A. P\ x\} \rangle$  show ?thesis by simp
next
  assume  $\exists X \in \{x \in A. P\ x\} // R. \textit{infinite } X$ 
  from this  $\langle \textit{infinite } \{x \in A. P\ x\} \rangle$  show ?thesis
  using  $\langle \bigwedge F. F \in \{x \in A. P\ x\} // R \implies \text{card } F = k \rangle$  card.infinite by auto
qed
qed

```

```

lemma card-equiv-class:
  assumes finite A
  assumes equiv A R
  shows  $\text{card } A = \text{sum } \text{card } (A // R)$ 
proof –
  have  $(\lambda x. \text{True})$  respects R by (simp add: congruentI)
  from  $\langle \textit{finite } A \rangle \langle \textit{equiv } A\ R \rangle$  this show ?thesis
  using card-equiv-class-restricted[where  $P = \lambda x. \text{True}$ ] by auto
qed

```

```

lemma card-equiv-class-same-size:
  assumes equiv A R
  assumes  $\bigwedge F. F \in A // R \implies \text{card } F = k$ 
  shows  $\text{card } A = k * \text{card } (A // R)$ 
proof –
  have  $(\lambda x. \text{True})$  respects R by (simp add: congruentI)
  from  $\langle \textit{equiv } A\ R \rangle \langle \bigwedge F. F \in A // R \implies \text{card } F = k \rangle$  this show ?thesis
  using card-equiv-class-restricted-same-size[where  $P = \lambda x. \text{True}$ ] by auto
qed

```

1.3 Additions to FuncSet Theory

```

lemma finite-same-card-bij-on-ext-funcset:
  assumes finite A finite B  $\text{card } A = \text{card } B$ 
  shows  $\exists f. f \in A \rightarrow_E B \wedge \text{bij-betw } f\ A\ B$ 
proof –
  from assms obtain f' where  $f': \text{bij-betw } f'\ A\ B$ 
  using finite-same-card-bij by auto
  define f where  $\bigwedge x. f\ x = (\text{if } x \in A \text{ then } f'\ x \text{ else undefined})$ 
  have  $f \in A \rightarrow_E B$ 
  using f' unfolding f-def by (auto simp add: bij-betwE)
  moreover have bij-betw f A B
proof –
  have bij-betw f' A B  $\longleftrightarrow$  bij-betw f A B
  unfolding f-def by (auto intro!: bij-betw-cong)
  from this  $\langle \text{bij-betw } f'\ A\ B \rangle$  show ?thesis by auto
qed
ultimately show ?thesis by auto

```


qed

lemma *card-extensional-funcset*:

assumes *finite A*

shows $\text{card } (A \rightarrow_E B) = \text{card } B \wedge \text{card } A$

using *assms* by (*simp add: card-PiE prod-constant*)

lemma *bij-betw-implies-inj-on-and-card-eq*:

assumes *finite B*

assumes $f \in A \rightarrow_E B$

shows $\text{bij-betw } f A B \longleftrightarrow \text{inj-on } f A \wedge \text{card } A = \text{card } B$

proof

assume *bij-betw f A B*

from *this* show $\text{inj-on } f A \wedge \text{card } A = \text{card } B$

by (*simp add: bij-betw-imp-inj-on bij-betw-same-card*)

next

assume $\text{inj-on } f A \wedge \text{card } A = \text{card } B$

from *this* have *inj-on f A* and $\text{card } A = \text{card } B$ by *auto*

from $\langle f \in A \rightarrow_E B \rangle$ have $f' A \subseteq B$ by *auto*

from $\langle \text{inj-on } f A \rangle$ have $\text{card } (f' A) = \text{card } A$ by (*simp add: card-image*)

from $\langle f' A \subseteq B \rangle \langle \text{card } A = \text{card } B \rangle$ *this* have $f' A = B$

by (*simp add: <finite B> card-subset-eq*)

from $\langle \text{inj-on } f A \rangle$ *this* show *bij-betw f A B* by (*rule bij-betw-imageI*)

qed

lemma *bij-betw-implies-surj-on-and-card-eq*:

assumes *finite A*

assumes $f \in A \rightarrow_E B$

shows $\text{bij-betw } f A B \longleftrightarrow f' A = B \wedge \text{card } A = \text{card } B$

proof

assume *bij-betw f A B*

show $f' A = B \wedge \text{card } A = \text{card } B$

using $\langle \text{bij-betw } f A B \rangle$ *bij-betw-imp-surj-on bij-betw-same-card* by *blast*

next

assume $f' A = B \wedge \text{card } A = \text{card } B$

from *this* have $f' A = B$ and $\text{card } A = \text{card } B$ by *auto*

from *this* have *inj-on f A*

by (*simp add: <finite A> inj-on-iff-eq-card*)

from *this* $\langle f' A = B \rangle$ show *bij-betw f A B* by (*rule bij-betw-imageI*)

qed

1.4 Additions to Permutations Theory

lemma

assumes $f \in A \rightarrow_E B$ $f' A = B$

assumes *p permutes B* ($\forall x. f' x = p (f x)$)

shows $(\lambda b. \{x \in A. f x = b\})' B = (\lambda b. \{x \in A. f' x = b\})' B$

proof

show $(\lambda b. \{x \in A. f x = b\})' B \subseteq (\lambda b. \{x \in A. f' x = b\})' B$

```

proof
  fix X
  assume  $X \in (\lambda b. \{x \in A. f x = b\}) ' B$ 
  from this obtain  $b$  where  $X\text{-eq}: X = \{x \in A. f x = b\}$  and  $b \in B$  by blast
  from assms(3, 4) have  $\bigwedge x. f x = b \longleftrightarrow f' x = p b$  by (metis permutes-def)
  from  $\langle p \text{ permutes } B \rangle$   $X\text{-eq}$  this have  $X = \{x \in A. f' x = p b\}$ 
  using Collect-cong by auto
  moreover from  $\langle b \in B \rangle \langle p \text{ permutes } B \rangle$  have  $p b \in B$ 
  by (simp add: permutes-in-image)
  ultimately show  $X \in (\lambda b. \{x \in A. f' x = b\}) ' B$  by blast
qed
next
show  $(\lambda b. \{x \in A. f' x = b\}) ' B \subseteq (\lambda b. \{x \in A. f x = b\}) ' B$ 
proof
  fix X
  assume  $X \in (\lambda b. \{x \in A. f' x = b\}) ' B$ 
  from this obtain  $b$  where  $X\text{-eq}: X = \{x \in A. f' x = b\}$  and  $b \in B$  by blast
  from assms(3, 4) have  $\bigwedge x. f' x = b \longleftrightarrow f x = \text{inv } p b$ 
  by (auto simp add: permutes-inverses(1, 2))
  from  $\langle p \text{ permutes } B \rangle$   $X\text{-eq}$  this have  $X = \{x \in A. f x = \text{inv } p b\}$ 
  using Collect-cong by auto
  moreover from  $\langle b \in B \rangle \langle p \text{ permutes } B \rangle$  have  $\text{inv } p b \in B$ 
  by (simp add: permutes-in-image permutes-inv)
  ultimately show  $X \in (\lambda b. \{x \in A. f x = b\}) ' B$  by blast
qed
qed

```

1.5 Additions to List Theory

The theorem *card-lists-length-eq* contains the superfluous assumption *finite* A . Here, we derive that fact without that unnecessary assumption.

lemma *lists-length-eq-Suc-eq-image-Cons*:

```

 $\{xs. \text{set } xs \subseteq A \wedge \text{length } xs = \text{Suc } n\} = (\lambda(x, xs). x\#\text{xs}) ' (A \times \{xs. \text{set } xs \subseteq A$ 
 $\wedge \text{length } xs = n\})$ 
  (is  $?A = ?B$ )

```

proof

show $?A \subseteq ?B$

proof

fix xs

assume $xs \in ?A$

from *this* **show** $xs \in ?B$ **by** (*cases xs*) *auto*

qed

next

show $?B \subseteq ?A$ **by** *auto*

qed

lemma *lists-length-eq-Suc-eq-empty-iff*:

```

 $\{xs. \text{set } xs \subseteq A \wedge \text{length } xs = \text{Suc } n\} = \{\}$   $\longleftrightarrow A = \{\}$ 

```

proof (*induct n*)

```

case 0
have {xs. set xs ⊆ A ∧ length xs = Suc 0} = {x#[] | x. x ∈ A}
proof
  show {[x] | x. x ∈ A} ⊆ {xs. set xs ⊆ A ∧ length xs = Suc 0} by auto
next
  show {xs. set xs ⊆ A ∧ length xs = Suc 0} ⊆ {[x] | x. x ∈ A}
  proof
    fix xs
    assume xs ∈ {xs. set xs ⊆ A ∧ length xs = Suc 0}
    from this have set xs ⊆ A ∧ length xs = Suc 0 by simp
    from this have ∃x. xs = [x] ∧ x ∈ A
      by (metis Suc-length-conv insert-subset length-0-conv list.set(2))
    from this show xs ∈ {[x] | x. x ∈ A} by simp
  qed
qed
then show ?case by simp
next
  case (Suc n)
  from this show ?case by (auto simp only: lists-length-eq-Suc-eq-image-Cons)
qed

lemma lists-length-eq-empty-iff:
  {xs. set xs ⊆ A ∧ length xs = n} = {} ↔ (A = {} ∧ n > 0)
proof (cases n)
  case 0
  then show ?thesis by auto
next
  case (Suc n)
  then show ?thesis by (auto simp only: lists-length-eq-Suc-eq-empty-iff)
qed

lemma finite-lists-length-eq-iff:
  finite {xs. set xs ⊆ A ∧ length xs = n} ↔ (finite A ∨ n = 0)
proof
  assume finite {xs. set xs ⊆ A ∧ length xs = n}
  from this show finite A ∨ n = 0
  proof (induct n)
  case 0
  then show ?case by simp
  next
  case (Suc n)
  have inj (λ(x, xs). x#xs)
  by (auto intro: inj-onI)
  from this Suc(2) have finite (A × {xs. set xs ⊆ A ∧ length xs = n})
  using finite-imageD inj-on-subset subset-UNIV lists-length-eq-Suc-eq-image-Cons[of
A n]
  by fastforce
  from this have finite A
  by (cases A = {})

```

```

      (auto simp only: lists-length-eq-eq-empty-iff dest: finite-cartesian-productD1)
    from this show ?case by auto
  qed
next
  assume finite A ∨ n = 0
  from this show finite {xs. set xs ⊆ A ∧ length xs = n}
    by (auto intro: finite-lists-length-eq)
qed

lemma card-lists-length-eq:
  shows card {xs. set xs ⊆ B ∧ length xs = n} = card B ^ n
proof cases
  assume finite B
  then show ?thesis by (rule card-lists-length-eq)
next
  assume infinite B
  then show ?thesis
  proof cases
    assume n = 0
    from this have {xs. set xs ⊆ B ∧ length xs = n} = {[]} by auto
    from this ⟨n = 0⟩ show ?thesis by simp
  next
    assume n ≠ 0
    from this ⟨infinite B⟩ have infinite {xs. set xs ⊆ B ∧ length xs = n}
      by (simp add: finite-lists-length-eq-iff)
    from this ⟨infinite B⟩ show ?thesis by auto
  qed
qed
qed

```

1.6 Additions to Disjoint Set Theory

```

lemma bij-betw-congI:
  assumes bij-betw f A A'
  assumes ∀ a ∈ A. f a = g a
  shows bij-betw g A A'
using assms bij-betw-cong by fastforce

```

```

lemma disjoint-family-onI[intro]:
  assumes ∧ m n. m ∈ S ⇒ n ∈ S ⇒ m ≠ n ⇒ A m ∩ A n = {}
  shows disjoint-family-on A S
using assms unfolding disjoint-family-on-def by simp

```

The following lemma is not needed for this development, but is useful and could be moved to Disjoint Set theory or Equiv Relation theory if translated from set partitions to equivalence relations.

```

lemma infinite-partition-on:
  assumes infinite A
  shows infinite {P. partition-on A P}
proof -

```

```

from ⟨infinite A⟩ obtain  $x$  where  $x \in A$ 
  by (meson finite.intros(1) finite-subset subsetI)
from ⟨infinite A⟩ have infinite  $(A - \{x\})$ 
  by (simp add: infinite-remove)
define singletons-except-one
  where singletons-except-one =  $(\lambda a'. (\lambda a. \text{if } a = a' \text{ then } \{a, x\} \text{ else } \{a\})) \text{ ' } (A - \{x\})$ 
have infinite (singletons-except-one '  $(A - \{x\})$ )
proof -
  have inj-on singletons-except-one  $(A - \{x\})$ 
    unfolding singletons-except-one-def by (rule inj-onI) auto
  from ⟨infinite  $(A - \{x\})$ ⟩ this show ?thesis
    using finite-imageD by blast
qed
moreover have singletons-except-one '  $(A - \{x\}) \subseteq \{P. \text{partition-on } A P\}$ 
proof
  fix  $P$ 
  assume  $P \in \text{singletons-except-one ' } (A - \{x\})$ 
  from this obtain  $a'$  where  $a' \in A - \{x\}$  and  $P: P = \text{singletons-except-one } a'$  by blast
  have partition-on  $A ((\lambda a. \text{if } a = a' \text{ then } \{a, x\} \text{ else } \{a\}) \text{ ' } (A - \{x\}))$ 
    using ⟨ $x \in A$ ⟩ ⟨ $a' \in A - \{x\}$ ⟩ by (auto intro: partition-onI)
  from this have partition-on  $A P$ 
    unfolding  $P$  singletons-except-one-def .
  from this show  $P \in \{P. \text{partition-on } A P\}$  ..
qed
ultimately show ?thesis by (simp add: infinite-super)
qed

```

```

lemma finitely-many-partition-on-iff:
  finite  $\{P. \text{partition-on } A P\} \longleftrightarrow$  finite  $A$ 
using finitely-many-partition-on infinite-partition-on by blast

```

1.7 Additions to Multiset Theory

```

lemma mset-set-subseteq-mset-set:
  assumes finite  $B$   $A \subseteq B$ 
  shows mset-set  $A \subseteq\#$  mset-set  $B$ 
proof -
  from ⟨ $A \subseteq B$ ⟩ ⟨finite  $B$ ⟩ have finite  $A$  using finite-subset by blast
  {
    fix  $x$ 
    have count (mset-set  $A$ )  $x \leq$  count (mset-set  $B$ )  $x$ 
      using ⟨finite  $A$ ⟩ ⟨finite  $B$ ⟩ ⟨ $A \subseteq B$ ⟩
      by (metis count-mset-set(1, 3) eq-iff subsetCE zero-le-one)
  }
  from this show mset-set  $A \subseteq\#$  mset-set  $B$ 
    using mset-subset-eqI by blast
qed

```

lemma *mset-set-set-mset*:
assumes $M \subseteq\# \text{mset-set } A$
shows $\text{mset-set } (\text{set-mset } M) = M$
proof –
{
 fix x
 from $\langle M \subseteq\# \text{mset-set } A \rangle$ **have** $\text{count } M x \leq \text{count } (\text{mset-set } A) x$
 by (*simp add: mset-subset-eq-count*)
 from this **have** $\text{count } (\text{mset-set } (\text{set-mset } M)) x = \text{count } M x$
 by (*metis count-eq-zero-iff count-greater-eq-one-iff count-mset-set dual-order.antisym dual-order.trans finite-set-mset*)
}
from this **show** *?thesis* **by** (*simp add: multiset-eq-iff*)
qed

lemma *mset-set-set-mset'*:
assumes $\forall x. \text{count } M x \leq 1$
shows $\text{mset-set } (\text{set-mset } M) = M$
proof –
{
 fix x
 from *assms* **have** $\text{count } M x = 0 \vee \text{count } M x = 1$ **by** (*auto elim: le-SucE*)
 from this **have** $\text{count } (\text{mset-set } (\text{set-mset } M)) x = \text{count } M x$
 by (*metis count-eq-zero-iff count-mset-set(1,3) finite-set-mset*)
}
from this **show** *?thesis* **by** (*simp add: multiset-eq-iff*)
qed

lemma *card-set-mset*:
assumes $M \subseteq\# \text{mset-set } A$
shows $\text{card } (\text{set-mset } M) = \text{size } M$
using *assms*
by (*metis mset-set-set-mset size-mset-set*)

lemma *card-set-mset'*:
assumes $\forall x. \text{count } M x \leq 1$
shows $\text{card } (\text{set-mset } M) = \text{size } M$
using *assms*
by (*metis mset-set-set-mset' size-mset-set*)

lemma *count-mset-set-leq*:
assumes *finite A*
shows $\text{count } (\text{mset-set } A) x \leq 1$
using *assms* **by** (*metis count-mset-set(1,3) eq-iff zero-le-one*)

lemma *count-mset-set-leq'*:
assumes *finite A*
shows $\text{count } (\text{mset-set } A) x \leq \text{Suc } 0$

using *assms count-mset-set-leq* by *fastforce*

lemma *msubset-mset-set-iff*:

assumes *finite A*

shows $set\text{-}mset\ M \subseteq A \wedge (\forall x. count\ M\ x \leq 1) \longleftrightarrow (M \subseteq\# mset\text{-}set\ A)$

proof

assume $set\text{-}mset\ M \subseteq A \wedge (\forall x. count\ M\ x \leq 1)$

from *this assms* show $M \subseteq\# mset\text{-}set\ A$

by (*metis count-inI count-mset-set(1) le0 mset-subset-eqI subsetCE*)

next

assume $M \subseteq\# mset\text{-}set\ A$

from *this assms* have $set\text{-}mset\ M \subseteq A$

using *mset-subset-eqD* by *fastforce*

moreover {

fix *x*

from $\langle M \subseteq\# mset\text{-}set\ A \rangle$ have $count\ M\ x \leq count\ (mset\text{-}set\ A)\ x$

by (*simp add: mset-subset-eq-count*)

from *this* $\langle finite\ A \rangle$ have $count\ M\ x \leq 1$

by (*meson count-mset-set-leq le-trans*)

}

ultimately show $set\text{-}mset\ M \subseteq A \wedge (\forall x. count\ M\ x \leq 1)$ by *simp*

qed

lemma *image-mset-fun-upd*:

assumes $x \notin\# M$

shows $image\text{-}mset\ (f(x := y))\ M = image\text{-}mset\ f\ M$

using *assms* by (*induct M*) *auto*

1.8 Additions to Number Partitions Theory

lemma *Partition-diag*:

shows $Partition\ n\ n = 1$

by (*cases n*) (*auto simp only: Partition-diag Partition.simps(1)*)

1.9 Cardinality Theorems with Iverson Function

definition *iverson* :: $bool \Rightarrow nat$

where

$iverson\ b = (if\ b\ then\ 1\ else\ 0)$

lemma *card-partition-on-size1-eq-iverson*:

assumes *finite A*

shows $card\ \{P. partition\text{-}on\ A\ P \wedge card\ P \leq k \wedge (\forall X \in P. card\ X = 1)\} = iverson\ (card\ A \leq k)$

proof (*cases card A ≤ k*)

case *True*

from *this* $\langle finite\ A \rangle$ show *?thesis*

unfolding *iverson-def*

using *card-partition-on-size1-eq-1* by *fastforce*

next

```

case False
from this (finite A) show ?thesis
  unfolding iverson-def
  using card-partition-on-size1-eq-0 by fastforce
qed

lemma card-number-partitions-with-only-parts-1:
  card {N. (∀ n. n ∈ # N → n = 1) ∧ number-partition n N ∧ size N ≤ x} =
  iverson (n ≤ x)
proof –
  show ?thesis
  proof cases
    assume n ≤ x
    from this show ?thesis
    using card-number-partitions-with-only-parts-1-eq-1
    unfolding iverson-def by auto
  next
    assume  $\neg n \leq x$ 
    from this show ?thesis
    using card-number-partitions-with-only-parts-1-eq-0
    unfolding iverson-def by auto
  qed
qed

end

```

2 Main Observations on Operations and Permutations

```

theory Twelvefold-Way-Core
imports Preliminaries
begin

```

2.1 Range Multiset

2.1.1 Existence of a Suitable Finite Function

```

lemma obtain-function:
  assumes finite A
  assumes size M = card A
  shows  $\exists f. \text{image-mset } f \text{ (mset-set } A) = M$ 
using assms
proof (induct arbitrary: M rule: finite-induct)
  case empty
  from this show ?case by simp
next
  case (insert x A)
  from insert(1,2,4) have size M > 0
  by (simp add: card-gt-0-iff)

```


from *this* **obtain** $y \in\# M$
using *gr0-implies-Suc size-eq-Suc-imp-elem* **by** *blast*
from *insert(1,2,4)* **this** **have** $\text{size } (M - \{\#y\}) = \text{card } A$
by (*simp add: Diff-insert-absorb card-Diff-singleton-if insertI1 size-Diff-submset*)
from *insert.hyps* **this** **obtain** f' **where** $\text{image-mset } f' (\text{mset-set } A) = M - \{\#y\}$ **by** *blast*
from *this* **have** $\text{image-mset } (f'(x := y)) (\text{mset-set } (\text{insert } x A)) = M$
using $\langle \text{finite } A \rangle \langle x \notin A \rangle \langle y \in\# M \rangle$ **by** (*simp add: image-mset-fun-upd*)
from *this* **show** *?case* **by** *blast*
qed

lemma *obtain-function-on-ext-funcset:*

assumes *finite A*
assumes $\text{size } M = \text{card } A$
shows $\exists f \in A \rightarrow_E \text{set-mset } M. \text{image-mset } f (\text{mset-set } A) = M$
proof –
obtain f **where** *range-eq-M: image-mset f (mset-set A) = M*
using *obtain-function* $\langle \text{finite } A \rangle \langle \text{size } M = \text{card } A \rangle$ **by** *blast*
let $?f = \lambda x. \text{if } x \in A \text{ then } f \text{ else undefined}$
have $?f \in A \rightarrow_E \text{set-mset } M$
using *range-eq-M* $\langle \text{finite } A \rangle$ **by** *auto*
moreover **have** $\text{image-mset } ?f (\text{mset-set } A) = M$
using *range-eq-M* $\langle \text{finite } A \rangle$ **by** (*auto intro: multiset.map-cong0*)
ultimately **show** *?thesis* **by** *auto*
qed

2.1.2 Existence of Permutation

lemma *image-mset-eq-implies-bij-betw:*

fixes $f :: 'a1 \Rightarrow 'b$ **and** $f' :: 'a2 \Rightarrow 'b$
assumes *finite A finite A'*
assumes *mset-eq: image-mset f (mset-set A) = image-mset f' (mset-set A')*
obtains *bij* **where** *bij-betw* $\text{bij } A A'$ **and** $\forall x \in A. f x = f' (\text{bij } x)$
proof –
from $\langle \text{finite } A \rangle$ **have** [*simp*]: $\text{finite } \{a \in A. f a = (b::'b)\}$ **for** b **by** *auto*
from $\langle \text{finite } A' \rangle$ **have** [*simp*]: $\text{finite } \{a \in A'. f' a = (b::'b)\}$ **for** b **by** *auto*
have $f' A = f' A'$
proof –
have $f' A = f' (\text{set-mset } (\text{mset-set } A))$ **using** $\langle \text{finite } A \rangle$ **by** *simp*
also **have** $\dots = f' (\text{set-mset } (\text{mset-set } A'))$
by (*metis mset-eq multiset.set-map*)
also **have** $\dots = f' A'$ **using** $\langle \text{finite } A' \rangle$ **by** *simp*
finally **show** *?thesis* .
qed

have $\forall b \in (f' A). \exists \text{bij}. \text{bij-betw } \text{bij } \{a \in A. f a = b\} \{a \in A'. f' a = b\}$

proof

fix b

from *mset-eq* **have**

$\text{count } (\text{image-mset } f (\text{mset-set } A)) b = \text{count } (\text{image-mset } f' (\text{mset-set } A')) b$

```

by simp
  from this have card {a ∈ A. f a = b} = card {a ∈ A'. f' a = b}
  using ⟨finite A⟩ ⟨finite A'⟩
  by (simp add: count-image-mset-eq-card-vimage)
  from this show ∃ bij. bij-betw bij {a ∈ A. f a = b} {a ∈ A'. f' a = b}
  by (intro finite-same-card-bij) simp-all
qed
from bchoice [OF this]
obtain bij where bij: ∀ b ∈ f ' A. bij-betw (bij b) {a ∈ A. f a = b} {a ∈ A'. f' a
= b}
  by auto
define bij' where bij' = (λ a. bij (f a) a)
have bij-betw bij' A A'
proof -
  have disjoint-family-on (λ i. {a ∈ A'. f' a = i}) (f ' A)
  unfolding disjoint-family-on-def by auto
  moreover have bij-betw (λ a. bij (f a) a) {a ∈ A. f a = b} {a ∈ A'. f' a = b}
if b: b ∈ f ' A for b
  using bij b by (subst bij-betw-cong[where g=bij b]) auto
  ultimately have bij-betw (λ a. bij (f a) a) (∪ b ∈ f ' A. {a ∈ A. f a = b}) (∪ b ∈ f
' A. {a ∈ A'. f' a = b})
  by (rule bij-betw-UNION-disjoint)
  moreover have (∪ b ∈ f ' A. {a ∈ A. f a = b}) = A by auto
  moreover have (∪ b ∈ f ' A. {a ∈ A'. f' a = b}) = A' using ⟨f ' A = f' ' A'⟩
by auto
  ultimately show bij-betw bij' A A'
  unfolding bij'-def by (subst bij-betw-cong[where g=(λ a. bij (f a) a)]) auto
qed
moreover from bij have ∀ x ∈ A. f x = f' (bij' x)
  unfolding bij'-def using bij-betwE by fastforce
  ultimately show ?thesis by (rule that)
qed

lemma image-mset-eq-implies-permutes:
  fixes f :: 'a ⇒ 'b
  assumes finite A
  assumes mset-eq: image-mset f (mset-set A) = image-mset f' (mset-set A)
  obtains p where p permutes A and ∀ x ∈ A. f x = f' (p x)
proof -
  from assms obtain b where bij-betw b A A and ∀ x ∈ A. f x = f' (b x)
  using image-mset-eq-implies-bij-betw by blast
  define p where p = (λ a. if a ∈ A then b a else a)
  have p permutes A
  proof (rule bij-imp-permutes)
    show bij-betw p A A
    unfolding p-def by (simp add: ⟨bij-betw b A A⟩ bij-betw-cong)
  next
  fix x
  assume x ∉ A

```

from *this* show $p\ x = x$
 unfolding *p-def* by *simp*
 qed
 moreover from $\langle \forall x \in A. f\ x = f'\ (b\ x) \rangle$ have $\forall x \in A. f\ x = f'\ (p\ x)$
 unfolding *p-def* by *simp*
 ultimately show *?thesis* by (rule *that*)
 qed

2.2 Domain Partition

2.2.1 Existence of a Suitable Finite Function

lemma *obtain-function-with-partition*:

assumes *finite A* *finite B*
 assumes *partition-on A P*
 assumes *card P ≤ card B*
 shows $\exists f \in A \rightarrow_E B. (\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\} = P$
 proof –
 obtain *g'* where *bij-betw g' P (g' ' P)* and $g' \text{ ' } P \subseteq B$
 by (*meson assms card-le-inj finite-elements inj-on-imp-bij-betw*)
 define *f* where $\bigwedge a. f\ a = (\text{if } a \in A \text{ then } g' \text{ (THE } X. a \in X \wedge X \in P) \text{ else undefined)}$
 have $f \in A \rightarrow_E B$
 unfolding *f-def*
 using $\langle g' \text{ ' } P \subseteq B \rangle$ *assms(3)* *partition-on-the-part-mem* by *fastforce*
 moreover have $(\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\} = P$
 proof
 show $(\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\} \subseteq P$
 proof
 fix *X*
 assume $X: X \in (\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\}$
 from *this* obtain *b* where $b \in B$ and $X = \{x' \in A. f\ x' = b\}$ by *auto*
 from *this* *X* obtain *a* where $a \in A$ and $a \in X$ and $f\ a = b$ by *blast*
 have $(\text{THE } X. a \in X \wedge X \in P) \in P$
 using $\langle a \in A \rangle$ $\langle \text{partition-on } A\ P \rangle$ by (*simp add: partition-on-the-part-mem*)
 from $\langle X = \{x' \in A. f\ x' = b\} \rangle$ have *X-eq1*: $X = \{x' \in A. g' \text{ (THE } X. x' \in X \wedge X \in P) = b\}$
 unfolding *f-def* by *auto*
 also have $\dots = \{x' \in A. (\text{THE } X. x' \in X \wedge X \in P) = \text{inv-into } P\ g'\ b\}$
 proof –
 {
 fix *x'*
 assume $x' \in A$
 have $(\text{THE } X. x' \in X \wedge X \in P) \in P$
 using $\langle \text{partition-on } A\ P \rangle$ $\langle x' \in A \rangle$ by (*simp add: partition-on-the-part-mem*)
 from *X-eq1* $\langle a \in X \rangle$ have $g' \text{ (THE } X. a \in X \wedge X \in P) = b$
 unfolding *f-def* by *auto*
 from *this* $\langle (\text{THE } X. a \in X \wedge X \in P) \in P \rangle$ have $b \in g' \text{ ' } P$ by *auto*
 have $(g' \text{ (THE } X. x' \in X \wedge X \in P) = b) \longleftrightarrow ((\text{THE } X. x' \in X \wedge X \in P) = \text{inv-into } P\ g'\ b)$
 }
 }

```

proof –
  from  $\langle (THE\ X.\ x' \in X \wedge X \in P) \in P \rangle$ 
  have  $(g' (THE\ X.\ x' \in X \wedge X \in P) = b) \longleftrightarrow (inv\text{-}into\ P\ g' (g' (THE\ X.\ x' \in X \wedge X \in P))) = inv\text{-}into\ P\ g' b)$ 
  using  $\langle b \in g' \text{' } P \rangle$  by (auto intro: inv-into-injective)
  moreover have  $inv\text{-}into\ P\ g' (g' (THE\ X.\ x' \in X \wedge X \in P)) = (THE\ X.\ x' \in X \wedge X \in P)$ 
  using  $\langle bij\text{-}betw\ g' P (g' \text{' } P) \rangle$   $\langle (THE\ X.\ x' \in X \wedge X \in P) \in P \rangle$ 
  by (simp add: bij-betw-inv-into-left)
  ultimately show ?thesis by simp
qed
}
from this show ?thesis by auto
qed
finally have  $X\text{-}eq: X = \{x' \in A.\ (THE\ X.\ x' \in X \wedge X \in P) = inv\text{-}into\ P\ g' b\}$ .
moreover have  $inv\text{-}into\ P\ g' b \in P$ 
proof –
  from  $X\text{-}eq$  have  $eq: inv\text{-}into\ P\ g' b = (THE\ X.\ a \in X \wedge X \in P)$ 
  using  $\langle a \in X \rangle$   $\langle a \in A \rangle$  by auto
  from this show ?thesis
  using  $\langle (THE\ X.\ a \in X \wedge X \in P) \in P \rangle$  by simp
qed
ultimately have  $X = inv\text{-}into\ P\ g' b$ 
  using partition-on-all-in-part-eq-part[OF  $\langle partition\text{-}on\ A\ P \rangle$ ] by blast
from this  $\langle inv\text{-}into\ P\ g' b \in P \rangle$  show  $X \in P$  by blast
qed
next
show  $P \subseteq (\lambda b.\ \{x \in A.\ f\ x = b\}) \text{' } B - \{\{\}\}$ 
proof
  fix  $X$ 
  assume  $X \in P$ 
  from assms(3) this have  $X \neq \{\}$ 
  by (auto elim: partition-onE)
  moreover have  $X \in (\lambda b.\ \{x \in A.\ f\ x = b\}) \text{' } B$ 
proof
  show  $g' X \in B$ 
  using  $\langle X \in P \rangle$   $\langle g' \text{' } P \subseteq B \rangle$  by blast
  show  $X = \{x \in A.\ f\ x = g' X\}$ 
proof
  show  $X \subseteq \{x \in A.\ f\ x = g' X\}$ 
proof
  fix  $x$ 
  assume  $x \in X$ 
  from this have  $x \in A$ 
  using  $\langle X \in P \rangle$  assms(3) by (fastforce elim: partition-onE)
  have  $(THE\ X.\ x \in X \wedge X \in P) = X$ 
  using  $\langle X \in P \rangle$   $\langle x \in X \rangle$  assms(3) partition-on-the-part-eq by fastforce
  from this  $\langle x \in A \rangle$  have  $f\ x = g' X$ 

```

```

      unfolding f-def by auto
      from this ⟨x ∈ A⟩ show x ∈ {x ∈ A. f x = g' X} by auto
    qed
  next
  show {x ∈ A. f x = g' X} ⊆ X
  proof
    fix x
    assume x ∈ {x ∈ A. f x = g' X}
    from this have x ∈ A and g-eq: g' (THE X. x ∈ X ∧ X ∈ P) = g' X
      unfolding f-def by auto
    from ⟨x ∈ A⟩ have (THE X. x ∈ X ∧ X ∈ P) ∈ P
      using assms(3) by (simp add: partition-on-the-part-mem)
    from this g-eq have (THE X. x ∈ X ∧ X ∈ P) = X
      using ⟨X ∈ P⟩ ⟨bij-betw g' P (g' ' P)⟩
      by (metis bij-betw-inv-into-left)
    from this ⟨x ∈ A⟩ assms(3) show x ∈ X
      using partition-on-in-the-unique-part by fastforce
    qed
  qed
  qed
  ultimately show X ∈ (λb. {x ∈ A. f x = b}) ' B - {{{}}
    by auto
  qed
  ultimately show ?thesis by blast
qed

```

2.2.2 Equality under Permutation Application

lemma *permutes-implies-inv-image-on-eq*:

```

  assumes p permutes B
  shows (λb. {x ∈ A. p (f x) = b}) ' B = (λb. {x ∈ A. f x = b}) ' B
proof -
  have ∀ b ∈ B. ∀ x ∈ A. p (f x) = b ⟷ f x = inv p b
    using ⟨p permutes B⟩ by (auto simp add: permutes-inverses)
  from this have (λb. {x ∈ A. p (f x) = b}) ' B = (λb. {x ∈ A. f x = inv p b}) '
  B
    using image-cong by blast
  also have ... = (λb. {x ∈ A. f x = b}) ' inv p ' B
    by (auto simp add: image-comp)
  also have ... = (λb. {x ∈ A. f x = b}) ' B
    by (simp add: ⟨p permutes B⟩ permutes-inv permutes-image)
  finally show ?thesis .
qed

```

2.2.3 Existence of Permutation

lemma *the-elem*:

```

  assumes f ∈ A →E B f' ∈ A →E B

```

assumes *partitions-eq*: $(\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$
assumes $x \in A$
shows *the-elem* $(f \text{ ' } \{x \in A. f' x = f' x\}) = f x$
proof –
from $\langle x \in A \rangle$ **have** $x: x \in \{x' \in A. f' x' = f' x\}$ **by** *blast*
have $f' x \in B$
using $\langle x \in A \rangle \langle f' \in A \rightarrow_E B \rangle$ **by** *blast*
from *this* **have** $\{x' \in A. f' x' = f' x\} \in (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$
using $\langle x \in A \rangle$ **by** *blast*
from *this* **have** $\{x' \in A. f' x' = f' x\} \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}$
using *partitions-eq* **by** *blast*
from *this* **obtain** b **where** *eq*: $\{x' \in A. f' x' = f' x\} = \{x' \in A. f x' = b\}$ **by** *blast*
also from x **this show** *the-elem* $(f \text{ ' } \{x' \in A. f' x' = f' x\}) = f x$
by (*metis (mono-tags, lifting) empty-iff mem-Collect-eq the-elem-image-unique*)
qed

lemma *the-elem-eq*:
assumes $f \in A \rightarrow_E B$
assumes $b \in f \text{ ' } A$
shows *the-elem* $(f \text{ ' } \{x' \in A. f x' = b\}) = b$
proof –
from $\langle b \in f \text{ ' } A \rangle$ **obtain** a **where** $a \in A$ **and** $b = f a$ **by** *blast*
from *this* **show** *the-elem* $(f \text{ ' } \{x' \in A. f x' = b\}) = b$
using *the-elem[OF $\langle f \in A \rightarrow_E B \rangle \langle f \in A \rightarrow_E B \rangle$]* **by** *simp*
qed

lemma *partitions-eq-implies*:
assumes $f \in A \rightarrow_E B$ $f' \in A \rightarrow_E B$
assumes *partitions-eq*: $(\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$
assumes $x \in A$ $x' \in A$
assumes $f x = f x'$
shows $f' x = f' x'$
proof –
have $f x \in B$ **and** $x \in \{a \in A. f a = f x\}$ **and** $x' \in \{a \in A. f a = f x\}$
using $\langle f \in A \rightarrow_E B \rangle \langle x \in A \rangle \langle x' \in A \rangle \langle f x = f x' \rangle$ **by** *auto*
moreover have $\{a \in A. f a = f x\} \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}$
using $\langle f x \in B \rangle \langle x \in \{a \in A. f a = f x\} \rangle$ **by** *auto*
ultimately obtain b **where** $x \in \{a \in A. f' a = b\}$ **and** $x' \in \{a \in A. f' a = b\}$
using *partitions-eq* **by** (*metis (no-types, lifting) Diff-iff imageE*)
from *this* **show** $f' x = f' x'$ **by** *auto*
qed

lemma *card-domain-partitions*:
assumes $f \in A \rightarrow_E B$
assumes *finite B*
shows *card* $((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) = \text{card} (f \text{ ' } A)$

proof –
note $[simp] = \text{the-elem-eq}[OF \langle f \in A \rightarrow_E B \rangle]$
have $\text{bij-betw } (\lambda X. \text{the-elem } (f \text{ ' } X)) ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) (f \text{ ' } A)$
proof (rule bij-betw-imageI)
show $\text{inj-on } (\lambda X. \text{the-elem } (f \text{ ' } X)) ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\})$
proof (rule inj-onI)
fix $X X'$
assume $X: X \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}$
assume $X': X' \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}$
assume $\text{eq}: \text{the-elem } (f \text{ ' } X) = \text{the-elem } (f \text{ ' } X')$
from X **obtain** b **where** $b \in B$ **and** $X\text{-eq}: X = \{x \in A. f x = b\}$ **by** blast
from X **this have** $b \in f \text{ ' } A$
using $\text{Collect-empty-eq Diff-iff image-iff insertCI}$ **by** auto
from X' **obtain** b' **where** $b' \in B$ **and** $X'\text{-eq}: X' = \{x \in A. f x = b'\}$ **by** blast
blast
from X' **this have** $b' \in f \text{ ' } A$
using $\text{Collect-empty-eq Diff-iff image-iff insertCI}$ **by** auto
from $X\text{-eq } X'\text{-eq eq} \langle \bigwedge b. b \in f \text{ ' } A \implies \text{the-elem } (f \text{ ' } \{x' \in A. f x' = b\}) = b \rangle$
 $\langle b \in f \text{ ' } A \rangle \langle b' \in f \text{ ' } A \rangle$
have $b = b'$ **by** auto
from **this show** $X = X'$
using $X\text{-eq } X'\text{-eq}$ **by** simp
qed
show $(\lambda X. \text{the-elem } (f \text{ ' } X)) \text{ ' } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) = f \text{ ' } A$
proof
show $(\lambda X. \text{the-elem } (f \text{ ' } X)) \text{ ' } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) \subseteq f \text{ ' } A$
using $\langle \bigwedge b. b \in f \text{ ' } A \implies \text{the-elem } (f \text{ ' } \{x' \in A. f x' = b\}) = b \rangle$ **by** auto
next
show $f \text{ ' } A \subseteq (\lambda X. \text{the-elem } (f \text{ ' } X)) \text{ ' } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\})$
proof
fix b
assume $b \in f \text{ ' } A$
from **this have** $b = \text{the-elem } (f \text{ ' } \{x \in A. f x = b\})$
using $\langle \bigwedge b. b \in f \text{ ' } A \implies \text{the-elem } (f \text{ ' } \{x' \in A. f x' = b\}) = b \rangle$ **by** auto
moreover from $\langle b \in f \text{ ' } A \rangle$ **have** $\{x \in A. f x = b\} \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}$
using $\langle f \in A \rightarrow_E B \rangle$ **by** auto
ultimately show $b \in (\lambda X. \text{the-elem } (f \text{ ' } X)) \text{ ' } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\})$..
qed
qed
qed
from **this show** $?thesis$ **by** (rule $\text{bij-betw-same-card}$)
qed

lemma $\text{partitions-eq-implies-permutes}$:
assumes $f \in A \rightarrow_E B$ $f' \in A \rightarrow_E B$
assumes $\text{finite } B$
assumes $\text{partitions-eq}: (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$

$= b\} \text{ ' } B - \{\{\}\}$
shows $\exists p. p \text{ permutes } B \wedge (\forall x \in A. f x = p (f' x))$
proof –
have *card-eq*: $\text{card } (f' \text{ ' } A) = \text{card } (f \text{ ' } A)$
using *card-domain-partitions*[*OF* $\langle f \in A \rightarrow_E B \rangle \langle \text{finite } B \rangle$]
using *card-domain-partitions*[*OF* $\langle f' \in A \rightarrow_E B \rangle \langle \text{finite } B \rangle$]
using *partitions-eq* **by** *simp*
have $f' \text{ ' } A \subseteq B \text{ ' } f \text{ ' } A \subseteq B$
using $\langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle$ **by** *auto*
from *this* *card-eq* **have** $\text{card } (B - f' \text{ ' } A) = \text{card } (B - f \text{ ' } A)$
using $\langle \text{finite } B \rangle$ **by** (*auto simp add: card-Diff-subset finite-subset*)
from *this* **obtain** p' **where** *bij-betw* $p' (B - f' \text{ ' } A) (B - f \text{ ' } A)$
using $\langle \text{finite } B \rangle$ **by** (*metis finite-same-card-bij finite-Diff*)
from *this* **have** $p' \text{ ' } (B - f' \text{ ' } A) = (B - f \text{ ' } A)$
by (*simp add: bij-betw-imp-surj-on*)
define p **where** $\bigwedge b. p b = (\text{if } b \in B \text{ then } (\text{if } b \in f' \text{ ' } A \text{ then the-elem } (f' \text{ ' } \{x \in A. f' x = b\}) \text{ else } p' b) \text{ else } b)$
have $\forall x \in A. f x = p (f' x)$
proof
fix x
assume $x \in A$
from *this* *partitions-eq* **have** *the-elem* $(f' \text{ ' } \{xa \in A. f' xa = f' x\}) = f x$
using *the-elem*[*OF* $\langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle$] **by** *auto*
from *this* **show** $f x = p (f' x)$
using $\langle x \in A \rangle$ *p-def* $\langle f' \in A \rightarrow_E B \rangle$ **by** *auto*
qed
moreover **have** $p \text{ permutes } B$
proof (*rule bij-imp-permutes*)
let $?invp = \lambda b. \text{if } b \in f' \text{ ' } A \text{ then the-elem } (f' \text{ ' } \{x \in A. f x = b\}) \text{ else } b$
note [*simp*] = *the-elem*[*OF* $\langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle$] *partitions-eq*
show *bij-betw* $p B B$
proof (*rule bij-betw-imageI*)
show $p \text{ ' } B = B$
proof
have $(\lambda b. \text{the-elem } (f' \text{ ' } \{x \in A. f' x = b\})) \text{ ' } (f' \text{ ' } A) \subseteq B$
using $\langle f \in A \rightarrow_E B \rangle$ **by** *auto*
from $\langle p' \text{ ' } (B - f' \text{ ' } A) = (B - f \text{ ' } A) \rangle$ *this* **show** $p \text{ ' } B \subseteq B$
unfolding *p-def* $\langle f \in A \rightarrow_E B \rangle$ **by** *force*
next
show $B \subseteq p \text{ ' } B$
proof
fix b
assume $b \in B$
show $b \in p \text{ ' } B$
proof (*cases* $b \in f' \text{ ' } A$)
assume $b \notin f' \text{ ' } A$
note $\langle p' \text{ ' } (B - f' \text{ ' } A) = (B - f \text{ ' } A) \rangle$
from *this* $\langle b \in B \rangle \langle b \notin f' \text{ ' } A \rangle$ **show** *?thesis*
unfolding *p-def* **by** *auto*


```

next
  assume  $b \in f' \text{ ' } A$ 
  from this  $\langle \forall x \in A. f \ x = p \ (f' \ x) \rangle \langle b \in B \rangle$  show ?thesis
  using  $\langle f' \in A \rightarrow_E \ B \rangle$  by auto
qed
qed
qed
next
show inj-on  $p \ B$ 
proof (rule inj-onI)
  fix  $b \ b'$ 
  assume  $b \in B \ b' \in B \ p \ b = p \ b'$ 
  have  $b \in f' \text{ ' } A \longleftrightarrow b' \in f' \text{ ' } A$ 
  proof -
    have  $b \in f' \text{ ' } A \longleftrightarrow p \ b \in f \text{ ' } A$ 
      unfolding p-def using  $\langle b \in B \rangle \langle p' \text{ ' } (B - f' \text{ ' } A) = B - f' \text{ ' } A \rangle$  by auto
    also have  $p \ b \in f \text{ ' } A \longleftrightarrow p \ b' \in f \text{ ' } A$ 
      using  $\langle p \ b = p \ b' \rangle$  by simp
    also have  $p \ b' \in f \text{ ' } A \longleftrightarrow b' \in f' \text{ ' } A$ 
      unfolding p-def using  $\langle b' \in B \rangle \langle p' \text{ ' } (B - f' \text{ ' } A) = B - f' \text{ ' } A \rangle$  by auto
    finally show ?thesis .
  qed
  from this have  $(b \in f' \text{ ' } A \wedge b' \in f' \text{ ' } A) \vee (b \notin f' \text{ ' } A \wedge b' \notin f' \text{ ' } A)$  by
blast
  from this show  $b = b'$ 
  proof
    assume  $b \in f' \text{ ' } A \wedge b' \in f' \text{ ' } A$ 
    from this obtain  $a \ a'$  where  $a \in A \ b = f' \ a$  and  $a' \in A \ b' = f' \ a'$  by
auto
    from this  $\langle b \in B \rangle \langle b' \in B \rangle$  have  $p \ b = f \ a \ p \ b' = f \ a'$ 
      unfolding p-def by auto
    from this  $\langle p \ b = p \ b' \rangle$  have  $f \ a = f \ a'$  by simp
    from this have  $f' \ a = f' \ a'$ 
  using partitions-eq-implies[OF  $\langle f \in A \rightarrow_E \ B \rangle \langle f' \in A \rightarrow_E \ B \rangle$  partitions-eq]
    using  $\langle a \in A \rangle \langle a' \in A \rangle$  by blast
    from this show  $b = b'$ 
    using  $\langle b' = f' \ a' \rangle \langle b = f' \ a \rangle$  by simp
  next
    assume  $b \notin f' \text{ ' } A \wedge b' \notin f' \text{ ' } A$ 
    from this  $\langle b \in B \rangle \langle b' \in B \rangle$  have  $p \ b' = p' \ b' \ p \ b = p' \ b$ 
      unfolding p-def by auto
    from this  $\langle p \ b = p \ b' \rangle$  have  $p' \ b = p' \ b'$  by simp
    moreover have  $b \in B - f' \text{ ' } A \ b' \in B - f' \text{ ' } A$ 
      using  $\langle b \in B \rangle \langle b' \in B \rangle \langle b \notin f' \text{ ' } A \wedge b' \notin f' \text{ ' } A \rangle$  by auto
    ultimately show  $b = b'$ 
    using bij-betw  $p' \ - \rightarrow$  by (metis bij-betw-inv-into-left)
  qed
qed
qed

```

```

next
  fix x
  assume  $x \notin B$ 
  from this show  $p\ x = x$ 
    using  $\langle f' \in A \rightarrow_E B \rangle$  p-def by auto
  qed
ultimately show ?thesis by blast
qed

```

2.3 Number Partition of Range

2.3.1 Existence of a Suitable Finite Function

lemma *obtain-partition*:

```

assumes finite A
assumes number-partition (card A) N
shows  $\exists P.$  partition-on A P  $\wedge$  image-mset card (mset-set P) = N
using assms

```

proof (induct N arbitrary: A)

```

case empty
from this have  $A = \{\}$ 
  unfolding number-partition-def by auto
from this have partition-on A  $\{\}$  by (simp add: partition-on-empty)
moreover have image-mset card (mset-set  $\{\}$ ) =  $\{\#\}$  by simp
ultimately show ?case by blast

```

next

```

case (add x N)
from add.prem(2) have  $0 \notin \#\$  add-mset x N and sum-mset (add-mset x N) =
card A
  unfolding number-partition-def by auto
from this have  $x \leq$  card A by auto
from this obtain X where  $X \subseteq A$  and card X = x
  using subset-with-given-card-exists by auto
from this have  $X \neq \{\}$ 
  using  $\langle 0 \notin \#\$  add-mset x N  $\rangle$   $\langle$  finite A  $\rangle$  by auto
have sum-mset N = card (A - X)
  using  $\langle$  sum-mset (add-mset x N) = card A  $\rangle$   $\langle$  card X = x  $\rangle$   $\langle$  X  $\subseteq$  A  $\rangle$ 
  by (metis add commute add.prem(1) add-diff-cancel-right' card-Diff-subset
infinite-super sum-mset.add-mset)
from this  $\langle 0 \notin \#\$  add-mset x N  $\rangle$  have number-partition (card (A - X)) N
  unfolding number-partition-def by auto
from this obtain P where partition-on (A - X) P and eq-N: image-mset card
(mset-set P) = N
  using add.hyps  $\langle$  finite A  $\rangle$  by auto
from  $\langle$  partition-on (A - X) P  $\rangle$  have finite P
  using  $\langle$  finite A  $\rangle$  finite-elements by blast
from  $\langle$  partition-on (A - X) P  $\rangle$  have  $X \notin P$ 
  using  $\langle$  X  $\neq \{\}$   $\rangle$  partition-onD1 by fastforce
have partition-on A (insert X P)
  using  $\langle$  partition-on (A - X) P  $\rangle$   $\langle$  X  $\subseteq$  A  $\rangle$   $\langle$  X  $\neq \{\}$   $\rangle$ 

```

by (*rule partition-on-insert'*)
moreover have *image-mset* card (*mset-set* (*insert X P*)) = *add-mset x N*
using *eq-N* $\langle \text{card } X = x \rangle \langle \text{finite } P \rangle \langle X \notin P \rangle$ **by** *simp*
ultimately show *?case* **by** *blast*
qed

lemma *obtain-extensional-function-from-number-partition:*

assumes *finite A finite B*
assumes *number-partition* (*card A*) *N*
assumes *size N ≤ card B*
shows $\exists f \in A \rightarrow_E B. \text{image-mset } (\lambda X. \text{card } X) (\text{mset-set } (((\lambda b. \{x \in A. f x = b\})) \text{ ' } B - \{\{\}\})) = N$
proof –
obtain *P* **where** *partition-on A P* **and** *eq-N: image-mset card (mset-set P) = N*
using *assms obtain-partition by blast*
from *eq-N[symmetric]* $\langle \text{size } N \leq \text{card } B \rangle$ **have** *card P ≤ card B* **by** *simp*
from $\langle \text{partition-on } A P \rangle$ **this** **obtain** *f* **where** $f \in A \rightarrow_E B$
and *eq-P: (λb. {x ∈ A. f x = b}) ' B - {{}} = P*
using *obtain-function-with-partition[OF <finite A> <finite B>]* **by** *blast*
have *image-mset (λX. card X) (mset-set (((λb. {x ∈ A. f x = b})) ' B - {{}}))*
 = *N*
using *eq-P eq-N* **by** *simp*
from *this* $\langle f \in A \rightarrow_E B \rangle$ **show** *?thesis* **by** *auto*
qed

2.3.2 Equality under Permutation Application

lemma *permutes-implies-multiset-of-partition-cards-eq:*

assumes p_A *permutes A* p_B *permutes B*
shows *image-mset card (mset-set ((λb. {x ∈ A. p_B (f' (p_A x)) = b}) ' B - {{}})) = image-mset card (mset-set ((λb. {x ∈ A. f' x = b}) ' B - {{}}))*
proof –
have *inj-on* ($\langle \text{inv } p_A \rangle$) ($\langle \lambda b. \{x \in A. f' x = b\} \text{ ' } B - \{\{\}\} \rangle$)
by (*meson* $\langle p_A \text{ permutes } A \rangle$ *inj-image-eq-iff inj-onI permutes-surj surj-imp-inj-inv*)
have *image-mset card (mset-set ((λb. {x ∈ A. p_B (f' (p_A x)) = b}) ' B - {{}}))*
 =
image-mset card (mset-set ((λX. inv p_A ' X) ' ((λb. {x ∈ A. f' x = b}) ' B - {{}}))
proof –
have $(\lambda b. \{x \in A. p_B (f' (p_A x)) = b\}) \text{ ' } B - \{\{\}\} = (\lambda b. \{x \in A. f' (p_A x) = b\}) \text{ ' } B - \{\{\}\}$
 = $(\lambda b. \text{inv } p_A \text{ ' } \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$
using *permutes-implies-inv-image-on-eq[OF <p_B permutes B>]* **by** *metis*
also have $\dots = (\lambda b. \text{inv } p_A \text{ ' } \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}$
proof –
have $\{x \in A. f' (p_A x) = b\} = \text{inv } p_A \text{ ' } \{x \in A. f' x = b\}$ **for** *b*
proof
show $\{x \in A. f' (p_A x) = b\} \subseteq \text{inv } p_A \text{ ' } \{x \in A. f' x = b\}$
proof

```

fix x
assume  $x \in \{x \in A. f' (p_A x) = b\}$ 
from this have  $x \in A f' (p_A x) = b$  by auto
  moreover from this  $\langle p_A \text{ permutes } A \rangle$  have  $p_A x \in A$  by (simp add:
permutes-in-image)
  moreover from  $\langle p_A \text{ permutes } A \rangle$  have  $x = \text{inv } p_A (p_A x)$ 
  using permutes-inverses(2) by fastforce
  ultimately show  $x \in \text{inv } p_A \{x \in A. f' x = b\}$  by auto
qed
next
show  $\text{inv } p_A \{x \in A. f' x = b\} \subseteq \{x \in A. f' (p_A x) = b\}$ 
proof
  fix x
  assume  $x \in \text{inv } p_A \{x \in A. f' x = b\}$ 
  from this obtain  $x'$  where  $x = \text{inv } p_A x'$   $x' \in A f' x' = b$  by auto
  from this  $\langle p_A \text{ permutes } A \rangle$  have  $x \in A$  by (simp add: permutes-in-image
permutes-inv)
  from  $\langle x = \text{inv } p_A x' \rangle \langle f' x' = b \rangle$  have  $f' (p_A x) = b$ 
  using  $\langle p_A \text{ permutes } A \rangle$  permutes-inverses(1) by fastforce
  from this  $\langle x \in A \rangle$  show  $x \in \{x \in A. f' (p_A x) = b\}$  by auto
qed
qed
from this show ?thesis by blast
qed
also have  $\dots = (\lambda X. \text{inv } p_A \{x \in A. f' x = b\}) \{B - \{\{\}\}\}$  by
auto
  finally show ?thesis by simp
qed
also have  $\dots = \text{image-mset } (\lambda X. \text{card } (\text{inv } p_A \{x \in A. f' x = b\}))$ 
 $(\text{mset-set } ((\lambda b. \{x \in A. f' x = b\}) \{B - \{\{\}\}\}))$ 
  using  $\langle \text{inj-on } ((\cdot) (\text{inv } p_A)) ((\lambda b. \{x \in A. f' x = b\}) \{B - \{\{\}\}\}) \rangle$ 
  by (simp only: image-mset-mset-set[symmetric] image-mset.compositionality)
(meson comp-apply)
also have  $\dots = \text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' x = b\}) \{B - \{\{\}\}\}))$ 
  using  $\langle p_A \text{ permutes } A \rangle$  by (simp add: card-image inj-on-inv-into permutes-surj)
  finally show ?thesis .
qed

```

2.3.3 Existence of Permutation

lemma *partition-implies-permutes:*

assumes *finite A*

assumes *partition-on A P partition-on A P'*

assumes *image-mset card (mset-set P') = image-mset card (mset-set P)*

obtains p **where** p *permutes* $A P' = (\lambda X. p \{X\} \{P\})$

proof –

from $\langle \text{partition-on } A P \rangle \langle \text{partition-on } A P' \rangle$ **have** *finite P finite P'*

using $\langle \text{finite } A \rangle$ *finite-elements* **by** *blast+*

from this $\langle \text{image-mset card } (\text{mset-set } P') = \text{image-mset card } (\text{mset-set } P) \rangle$

obtain *bij* **where** *bij-betw bij P P'* **and** $\forall X \in P. \text{card } X = \text{card } (\text{bij } X)$
using *image-mset-eq-implies-bij-betw* **by** *metis*
have $\forall X \in P. \exists p'. \text{bij-betw } p' X (\text{bij } X)$
proof
fix *X*
assume $X \in P$
from *this* **have** $X \subseteq A$
using $\langle \text{partition-on } A P \rangle \text{partition-onD1}$ **by** *fastforce*
from *this* **have** *finite X*
using $\langle \text{finite } A \rangle \text{rev-finite-subset}$ **by** *blast*
from $\langle X \in P \rangle$ **have** $\text{bij } X \in P'$
using $\langle \text{bij-betw bij } P P' \rangle \text{bij-betwE}$ **by** *blast*
from *this* **have** $\text{bij } X \subseteq A$
using $\langle \text{partition-on } A P' \rangle \text{partition-onD1}$ **by** *fastforce*
from *this* **have** *finite (bij X)*
using $\langle \text{finite } A \rangle \text{rev-finite-subset}$ **by** *blast*
from $\langle X \in P \rangle$ **have** $\text{card } X = \text{card } (\text{bij } X)$
using $\langle \forall X \in P. \text{card } X = \text{card } (\text{bij } X) \rangle$ **by** *blast*
from *this* **show** $\exists p'. \text{bij-betw } p' X (\text{bij } X)$
using $\langle \text{finite } (\text{bij } X) \rangle \langle \text{finite } X \rangle \text{finite-same-card-bij}$ **by** *blast*
qed
from *this* **have** $\exists p'. \forall X \in P. \text{bij-betw } (p' X) X (\text{bij } X)$ **by** *metis*
from *this* **obtain** *p'* **where** $p': \forall X \in P. \text{bij-betw } (p' X) X (\text{bij } X) \dots$
define *p* **where** $\bigwedge a. p a = (\text{if } a \in A \text{ then } p' (\text{THE } X. a \in X \wedge X \in P) a \text{ else } a)$
have *p* *permutes A*
proof –
have *bij-betw p A A*
proof –
have *disjoint-family-on bij P*
proof
fix *X X'*
assume $XX': X \in P X' \in P X \neq X'$
from *this* **have** $\text{bij } X \in P' \text{bij } X' \in P'$
using $\langle \text{bij-betw bij } P P' \rangle \text{bij-betwE}$ **by** *blast+*
moreover from XX' **have** $\text{bij } X \neq \text{bij } X'$
using $\langle \text{bij-betw bij } P P' \rangle$ **by** *(metis bij-betw-inv-into-left)*
ultimately show $\text{bij } X \cap \text{bij } X' = \{\}$
using $\langle \text{partition-on } A P' \rangle$ **by** *(meson partition-onE)*
qed
moreover have $\text{bij-betw } (\lambda a. p' (\text{THE } X. a \in X \wedge X \in P) a) X (\text{bij } X)$ **if**
 $X \in P$ **for** *X*
proof –
from $\langle X \in P \rangle$ **have** $\text{bij-betw } (p' X) X (\text{bij } X)$
using $\langle \forall X \in P. \text{bij-betw } (p' X) X (\text{bij } X) \rangle$ **by** *blast*
moreover from $\langle X \in P \rangle$ **have** $\forall a \in X. (\text{THE } X. a \in X \wedge X \in P) = X$
using $\langle \text{partition-on } A P \rangle \text{partition-on-the-part-eq}$ **by** *fastforce*
ultimately show *?thesis* **by** *(auto intro: bij-betw-congI)*
qed

ultimately have *bij-betw* $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) (\bigcup X \in P. X)$
 $(\bigcup X \in P. bij X)$
by *(rule bij-betw-UNION-disjoint)*
moreover have $(\bigcup X \in P. X) = A (\bigcup X \in P'. X) = A$
using $\langle partition-on A P \rangle \langle partition-on A P' \rangle partition-onD1$ **by** *auto*
moreover have $(\bigcup X \in P. bij X) = (\bigcup X \in P'. X)$
using $\langle bij-betw bij P P' \rangle bij-betw-imp-surj-on$ **by** *force*
ultimately have *bij-betw* $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) A A$ **by** *simp*
moreover have $\forall a \in A. p' (THE X. a \in X \wedge X \in P) a = p a$
unfolding *p-def* **by** *auto*
ultimately show *?thesis* **by** *(rule bij-betw-congI)*
qed
moreover have $p x = x$ **if** $x \notin A$ **for** x
using $\langle x \notin A \rangle p-def$ **by** *auto*
ultimately show *?thesis* **by** *(rule bij-imp-permutes)*
qed
moreover have $P' = (\lambda X. p ' X) ' P$
proof
show $P' \subseteq (\lambda X. p ' X) ' P$
proof
fix X
assume $X \in P'$
have *in-P: the-inv-into P bij X* $\in P$
using $\langle X \in P' \rangle \langle bij-betw bij P P' \rangle bij-betwE bij-betw-the-inv-into$ **by** *blast*
have *eq-X: bij (the-inv-into P bij X) = X*
using $\langle X \in P' \rangle \langle bij-betw bij P P' \rangle$
by *(meson f-the-inv-into-f-bij-betw)*
have $X = p ' (the-inv-into P bij X)$
proof
from *in-P* **have** *the-inv-into P bij X* $\subseteq A$
using $\langle partition-on A P \rangle partition-onD1$ **by** *fastforce*
have $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) ' the-inv-into P bij X = X$
proof
show $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) ' the-inv-into P bij X \subseteq X$
proof
fix x
assume $x \in (\lambda a. p' (THE X. a \in X \wedge X \in P) a) ' the-inv-into P bij X$
from *this* **obtain** a **where** *a-in: a* $\in the-inv-into P bij X$
and *x-eq: x = p' (THE X. a* $\in X \wedge X \in P) a$ **by** *blast*
have $(THE X. a \in X \wedge X \in P) = the-inv-into P bij X$
using *a-in in-P* $\langle partition-on A P \rangle partition-on-the-part-eq$
by *fastforce*
from *this x-eq* **have** *x-eq: x = p' (the-inv-into P bij X) a*
by *auto*
from *this* **have** $x \in bij (the-inv-into P bij X)$
using *a-in in-P bij-betwE p'* **by** *blast*
from *this eq-X* **show** $x \in X$ **by** *blast*
qed
next

```

show  $X \subseteq (\lambda a. p' (THE X. a \in X \wedge X \in P) a)$  ‘ the-inv-into P bij X
proof
  fix  $x$ 
  assume  $x \in X$ 
  let  $?X' = \text{the-inv-into } P \text{ bij } X$ 
  define  $x'$  where  $x' = \text{the-inv-into } ?X' (p' ?X') x$ 
  from in-P p' eq-X have bij-betw: bij-betw  $(p' ?X') ?X' X$  by auto
  from bij-betw  $\langle x \in X \rangle$  have  $x' \in ?X'$ 
    unfolding x'-def
    using bij-betwE bij-betw-the-inv-into by blast
  from this in-P have  $(THE X. x' \in X \wedge X \in P) = ?X'$ 
    using partition-on A P partition-on-the-part-eq by fastforce
  from this  $\langle x \in X \rangle$  have  $x = p' (THE X. x' \in X \wedge X \in P) x'$ 
    unfolding x'-def
    using bij-betw f-the-inv-into-f-bij-betw by fastforce
  from this  $\langle x' \in ?X' \rangle$  show  $x \in (\lambda a. p' (THE X. a \in X \wedge X \in P) a)$  ‘
the-inv-into P bij X ..
  qed
qed
from this  $\langle \text{the-inv-into } P \text{ bij } X \subseteq A \rangle$  show  $X \subseteq p'$  ‘ the-inv-into P bij X
  unfolding p-def by auto
next
show  $p'$  ‘ the-inv-into P bij X  $\subseteq X$ 
proof
  fix  $x$ 
  assume  $x \in p'$  ‘ the-inv-into P bij X
  from this obtain  $x'$  where  $x = p' x'$  and  $x' \in \text{the-inv-into } P \text{ bij } X$ 
    by auto
  have  $x' \in A$ 
    using  $\langle x' \in \text{the-inv-into } P \text{ bij } X \rangle$  assms(2) in-P partition-onD1 by
fastforce
  have eq:  $(THE X. x' \in X \wedge X \in P) = \text{the-inv-into } P \text{ bij } X$ 
    using  $\langle x' \in \text{the-inv-into } P \text{ bij } X \rangle$  assms(2) in-P partition-on-the-part-eq
by fastforce
  have  $p'$ :  $p' (\text{the-inv-into } P \text{ bij } X) x' \in X$ 
    using  $\langle x' \in \text{the-inv-into } P \text{ bij } X \rangle$  bij-betwE eq-X in-P p' by blast
  from  $\langle x = p' x' \rangle \langle x' \in A \rangle$  eq p' show  $x \in X$ 
    unfolding p-def by auto
  qed
qed
moreover from  $\langle X \in P' \rangle \langle \text{bij-betw } P P' \rangle$  have the-inv-into P bij X  $\in P$ 
  using bij-betwE bij-betw-the-inv-into by blast
ultimately show  $X \in (\lambda X. p' X)$  ‘  $P ..$ 
qed
next
show  $(\lambda X. p' X)$  ‘  $P \subseteq P'$ 
proof
  fix  $X'$ 
  assume  $X' \in (\lambda X. p' X)$  ‘  $P$ 

```

```

from this obtain  $X$  where  $X'$ -eq:  $X' = p \text{ ' } X$  and  $X \in P$  ..
from  $\langle X \in P \rangle$  have  $X \subseteq A$ 
  using assms(2) partition-onD1 by force
from  $\langle X \in P \rangle$   $p'$  have bij: bij-betw ( $p' X$ )  $X$  (bij X) by auto
have  $p \text{ ' } X \in P'$ 
proof –
  from  $\langle X \in P \rangle$   $\langle$ bij-betw bij P P' $\rangle$  have  $\text{bij } X \in P'$ 
    using bij-betwE by blast
  moreover have  $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X = \text{bij } X$ 
  proof
    show  $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X \subseteq \text{bij } X$ 
    proof
      fix  $x'$ 
      assume  $x' \in (\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X$ 
      from this obtain  $x$  where  $x \in X$  and  $x'$ -eq:  $x' = p' (THE X. x \in X \wedge$ 
 $X \in P) x$  ..
      from  $\langle X \in P \rangle$   $\langle x \in X \rangle$  have eq-X: (THE X. x \in X \wedge X \in P) = X
        using assms(2) partition-on-the-part-eq by fastforce
      from bij  $\langle x \in X \rangle$   $x'$ -eq eq-X show  $x' \in \text{bij } X$ 
        using bij-betwE by blast
    qed
  next
  show  $\text{bij } X \subseteq (\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X$ 
  proof
    fix  $x'$ 
    assume  $x' \in \text{bij } X$ 
    let  $?x = \text{inv-into } X (p' X) x'$ 
    from  $\langle x' \in \text{bij } X \rangle$  bij have  $?x \in X$ 
      by (metis bij-betw-imp-surj-on inv-into-into)
    from this  $\langle X \in P \rangle$  have  $(THE X. ?x \in X \wedge X \in P) = X$ 
      using assms(2) partition-on-the-part-eq by fastforce
    from this  $\langle x' \in \text{bij } X \rangle$  bij have  $x' = p' (THE X. ?x \in X \wedge X \in P) ?x$ 
      using bij-betw-inv-into-right by fastforce
    moreover from  $\langle x' \in \text{bij } X \rangle$  bij have  $?x \in X$ 
      by (metis bij-betw-imp-surj-on inv-into-into)
    ultimately show  $x' \in (\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X$  ..
  qed
qed
  ultimately have  $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X \in P'$  by simp
  have  $(\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X = (\lambda a. \text{if } a \in A \text{ then } p'$ 
 $(THE X. a \in X \wedge X \in P) a$  else  $a) \text{ ' } X$ 
    using  $\langle X \subseteq A \rangle$  by (auto intro: image-cong)
  from this show ?thesis
  using  $\langle (\lambda a. p' (THE X. a \in X \wedge X \in P) a) \text{ ' } X \in P' \rangle$  unfolding p-def
by auto
  qed
  from this  $X'$ -eq show  $X' \in P'$  by simp
  qed
qed

```


ultimately show *thesis* using *that* by *blast*
qed

lemma *permutes-domain-partition-eq*:

assumes $f \in A \rightarrow B$

assumes p_A permutes A

assumes $b \in B$

shows $p_A \langle \{x \in A. f x = b\} = \{x \in A. f (inv\ p_A\ x) = b\}$

proof

show $p_A \langle \{x \in A. f x = b\} \subseteq \{x \in A. f (inv\ p_A\ x) = b\}$

using $\langle p_A$ permutes $A \rangle$ *permutes-in-image permutes-inverses(2)* by *fastforce*

next

show $\{x \in A. f (inv\ p_A\ x) = b\} \subseteq p_A \langle \{x \in A. f x = b\}$

proof

fix x

assume $x \in \{x \in A. f (inv\ p_A\ x) = b\}$

from *this* **have** $x \in A$ $f (inv\ p_A\ x) = b$ by *auto*

from $\langle x \in A \rangle$ **have** $x = p_A (inv\ p_A\ x)$

using $\langle p_A$ permutes $A \rangle$ *permutes-inverses(1)* by *fastforce*

moreover from $\langle f (inv\ p_A\ x) = b \rangle$ $\langle x \in A \rangle$ **have** $inv\ p_A\ x \in \{x \in A. f x = b\}$

by (*simp add: $\langle p_A$ permutes $A \rangle$ permutes-in-image permutes-inv*)

ultimately show $x \in p_A \langle \{x \in A. f x = b\} ..$

qed

qed

lemma *image-domain-partition-eq*:

assumes $f \in A \rightarrow_E B$

assumes p_A permutes A

shows $(\lambda X. p_A \langle X \rangle \langle (\lambda b. \{x \in A. f x = b\}) \langle B \rangle = (\lambda b. \{x \in A. f (inv\ p_A\ x) = b\}) \langle B$

proof

from $\langle f \in A \rightarrow_E B \rangle$ **have** $f \in A \rightarrow B$ by *auto*

note $eq =$ *permutes-domain-partition-eq[OF $\langle f \in A \rightarrow B \rangle \langle p_A$ permutes $A \rangle$*

show $(\lambda X. p_A \langle X \rangle \langle (\lambda b. \{x \in A. f x = b\}) \langle B \rangle \subseteq (\lambda b. \{x \in A. f (inv\ p_A\ x) = b\}) \langle B$

proof

fix X

assume $X \in (\lambda X. p_A \langle X \rangle \langle (\lambda b. \{x \in A. f x = b\}) \langle B$

from *this* **obtain** $b \in B$ **and** X -*eq*: $X = p_A \langle \{x \in A. f x = b\}$ by

auto

from *this eq* **have** $X = \{x \in A. f (inv\ p_A\ x) = b\}$ by *simp*

from *this $\langle b \in B \rangle$* **show** $X \in (\lambda b. \{x \in A. f (inv\ p_A\ x) = b\}) \langle B ..$

qed

next

from $\langle f \in A \rightarrow_E B \rangle$ **have** $f \in A \rightarrow B$ by *auto*

note $eq =$ *permutes-domain-partition-eq[OF $\langle f \in A \rightarrow B \rangle \langle p_A$ permutes $A \rangle$, symmetric]*

show $(\lambda b. \{x \in A. f (inv\ p_A\ x) = b\}) \langle B \subseteq (\lambda X. p_A \langle X \rangle \langle (\lambda b. \{x \in A. f x = b\}) \langle B$

proof
fix X
assume $X \in (\lambda b. \{x \in A. f (inv\ p_A\ x) = b\}) \text{ ' } B$
from this obtain b **where** $b \in B$ **and** $X\text{-eq: } X = \{x \in A. f (inv\ p_A\ x) = b\}$
by *auto*
from this eq have $X = p_A \text{ ' } \{x \in A. f\ x = b\}$ **by** *simp*
from this $\langle b \in B \rangle$ **show** $X \in (\lambda X. p_A \text{ ' } X) \text{ ' } (\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B$ **by**
auto
qed
qed

lemma *multiset-of-partition-cards-eq-implies-permutes*:

assumes *finite* A *finite* B $f \in A \rightarrow_E B$ $f' \in A \rightarrow_E B$
assumes *eq*: *image-mset card* (*mset-set* (($\lambda b. \{x \in A. f\ x = b\}$) $\text{ ' } B - \{\{\}\}$)) =
image-mset card (*mset-set* (($\lambda b. \{x \in A. f'\ x = b\}$) $\text{ ' } B - \{\{\}\}$))
obtains $p_A\ p_B$ **where** p_A *permutes* A p_B *permutes* B $\forall x \in A. f\ x = p_B (f' (p_A\ x))$

proof –

have *partition-on* A (($\lambda b. \{x \in A. f\ x = b\}$) $\text{ ' } B - \{\{\}\}$)
using $\langle f \in A \rightarrow_E B \rangle$ **by** (*auto intro!*: *partition-onI*)
moreover have *partition-on* A (($\lambda b. \{x \in A. f'\ x = b\}$) $\text{ ' } B - \{\{\}\}$)
using $\langle f' \in A \rightarrow_E B \rangle$ **by** (*auto intro!*: *partition-onI*)
moreover note *partition-implies-permutes*[*OF* $\langle \text{finite } A \rangle$ - - *eq*]
ultimately obtain p_A **where** p_A *permutes* A **and**
inv-image-eq: ($\lambda b. \{x \in A. f\ x = b\}$) $\text{ ' } B - \{\{\}\}$ =
 (\cdot) $p_A \text{ ' } ((\lambda b. \{x \in A. f'\ x = b\}) \text{ ' } B - \{\{\}\})$ **by** *blast*
from $\langle p_A \text{ permutes } A \rangle$ **have** *inj* ((\cdot) p_A)
by (*meson injI inj-image-eq-iff permutes-inj*)
have *inv-image-eq'*: ($\lambda b. \{x \in A. f\ x = b\}$) $\text{ ' } B - \{\{\}\}$ = ($\lambda b. \{x \in A. f' (inv\ p_A\ x) = b\}$) $\text{ ' } B - \{\{\}\}$

proof –

note *inv-image-eq*
also have ($\lambda X. p_A \text{ ' } X$) $\text{ ' } ((\lambda b. \{x \in A. f'\ x = b\}) \text{ ' } B - \{\{\}\})$ = ($\lambda b. \{x \in A. f' (inv\ p_A\ x) = b\}$) $\text{ ' } B - \{\{\}\}$
using *image-domain-partition-eq*[*OF* $\langle f' \in A \rightarrow_E B \rangle \langle p_A \text{ permutes } A \rangle$]
by (*simp add: image-set-diff*[*OF* $\langle inj ((\cdot) p_A) \rangle$])
finally show *?thesis* .

qed

from $\langle p_A \text{ permutes } A \rangle$ **have** *inv* p_A *permutes* A
using *permutes-inv* **by** *blast*
have ($\lambda x. f' (inv\ p_A\ x) \in A \rightarrow_E B$)
using $\langle f' \in A \rightarrow_E B \rangle \langle inv\ p_A \text{ permutes } A \rangle$ *permutes-in-image* **by** *fastforce*
from $\langle f \in A \rightarrow_E B \rangle$ *this* $\langle \text{finite } B \rangle$ **obtain** p_B
where p_B *permutes* B **and** *eq''*: $\forall x \in A. f\ x = p_B (f' (inv\ p_A\ x))$
using *partitions-eq-implies-permutes*[*OF* - - - *inv-image-eq'*] **by** *blast*
from $\langle inv\ p_A \text{ permutes } A \rangle \langle p_B \text{ permutes } B \rangle$ *eq''* **that show** *thesis* **by** *blast*
qed

2.4 Bijections on Same Domain and Range

2.4.1 Existence of Domain Permutation

lemma *obtain-domain-permutation-for-two-bijections:*

assumes *bij-betw* f A B *bij-betw* f' A B

obtains p **where** p *permutes* A **and** $\forall a \in A. f\ a = f'\ (p\ a)$

proof –

let $?p = \lambda a. \text{if } a \in A \text{ then the-inv-into } A\ f'\ (f\ a) \text{ else } a$

have $?p$ *permutes* A

proof (*rule bij-imp-permutes*)

show *bij-betw* $?p$ A A

proof (*rule bij-betw-imageI*)

show *inj-on* $?p$ A

proof (*rule inj-onI*)

fix $a\ a'$

assume $a \in A\ a' \in A\ ?p\ a = ?p\ a'$

from *this* **have** *the-inv-into* $A\ f'\ (f\ a) = \text{the-inv-into } A\ f'\ (f\ a')$

using $\langle a \in A \rangle\ \langle a' \in A \rangle$ **by** *simp*

from *this* **have** $f\ a = f\ a'$

using $\langle a \in A \rangle\ \langle a' \in A \rangle$ *assms*

by (*metis bij-betwE f-the-inv-into-f-bij-betw*)

from *this* **show** $a = a'$

using $\langle a \in A \rangle\ \langle a' \in A \rangle$ *assms*

by (*metis bij-betw-inv-into-left*)

qed

next

show $?p\ 'A = A$

proof

show $?p\ 'A \subseteq A$

proof

fix a

assume $a \in ?p\ 'A$

from *this* **obtain** a' **where** $a' \in A$ **and** $a = \text{the-inv-into } A\ f'\ (f\ a')$ **by**

auto

from *this* *assms* **show** $a \in A$

by (*metis bij-betwE bij-betw-imp-inj-on bij-betw-imp-surj-on subset-iff the-inv-into-into*)

qed

next

show $A \subseteq ?p\ 'A$

proof

fix a

assume $a \in A$

from *this* *assms* **have** *the-inv-into* $A\ f\ (f'\ a) \in A$

by (*meson bij-betwE bij-betw-the-inv-into*)

moreover **from** $\langle a \in A \rangle$ *assms* **have** $a = \text{the-inv-into } A\ f'\ (f\ (\text{the-inv-into } A\ f\ (f'\ a)))$

by (*metis bij-betwE bij-betw-imp-inj-on f-the-inv-into-f-bij-betw the-inv-into-f-eq*)

ultimately **show** $a \in ?p\ 'A$ **by** *auto*

```

      qed
    qed
  qed
next
  fix a
  assume a ∉ A
  from this show ?p a = a by auto
qed
moreover have ∀ a ∈ A. f a = f' (?p a)
  using ⟨bij-betw f A B⟩ ⟨bij-betw f' A B⟩
  using bij-betwE f-the-inv-into-f-bij-betw by fastforce
moreover note that
ultimately show thesis by auto
qed

```

2.4.2 Existence of Range Permutation

lemma *obtain-range-permutation-for-two-bijections:*

```

  assumes bij-betw f A B bij-betw f' A B
  obtains p where p permutes B and ∀ a ∈ A. f a = p (f' a)

```

proof –

```

let ?p = λb. if b ∈ B then f (inv-into A f' b) else b

```

```

have ?p permutes B

```

```

proof (rule bij-imp-permutes)

```

```

  show bij-betw ?p B B

```

```

proof (rule bij-betw-imageI)

```

```

  show inj-on ?p B

```

```

proof (rule inj-onI)

```

```

  fix b b'

```

```

  assume b ∈ B b' ∈ B ?p b = ?p b'

```

```

  from this have f (inv-into A f' b) = f (inv-into A f' b')

```

```

    using ⟨b ∈ B⟩ ⟨b' ∈ B⟩ by simp

```

```

  from this have inv-into A f' b = inv-into A f' b'

```

```

    using ⟨b ∈ B⟩ ⟨b' ∈ B⟩ assms

```

```

    by (metis bij-betw-imp-surj-on bij-betw-inv-into-left inv-into-into)

```

```

  from this show b = b'

```

```

    using ⟨b ∈ B⟩ ⟨b' ∈ B⟩ assms(2)

```

```

    by (metis bij-betw-inv-into-right)

```

```

  qed

```

```

next

```

```

  show ?p ' B = B

```

```

proof

```

```

  from assms show ?p ' B ⊆ B

```

```

    by (auto simp add: bij-betwE bij-betw-def inv-into-into)

```

```

next

```

```

  show B ⊆ ?p ' B

```

```

proof

```

```

  fix b

```

```

  assume b ∈ B

```

```

    from this assms have  $f' (inv\text{-}into\ A\ f\ b) \in B$ 
      by (metis bij-betwE bij-betw-imp-surj-on inv-into-into)
    moreover have  $b = ?p (f' (inv\text{-}into\ A\ f\ b))$ 
      using assms  $\langle f' (inv\text{-}into\ A\ f\ b) \in B \rangle \langle b \in B \rangle$ 
    by (auto simp add: bij-betw-imp-surj-on bij-betw-inv-into-left bij-betw-inv-into-right
    inv-into-into)
    ultimately show  $b \in ?p ` B$  by auto
  qed
qed
qed
next
fix  $b$ 
assume  $b \notin B$ 
from this show  $?p\ b = b$  by auto
qed
moreover have  $\forall a \in A. f\ a = ?p (f'\ a)$ 
  using  $\langle \textit{bij-betw}\ f'\ A\ B \rangle$  bij-betw-inv-into-left bij-betwE by fastforce
moreover note that
ultimately show thesis by auto
qed
end

```

3 Definition of Equivalence Classes

```

theory Equiv-Relations-on-Functions
imports
  Preliminaries
  Twelvefold-Way-Core
begin

```

3.1 Permutation on the Domain

```

definition domain-permutation

```

```

where

```

```

  domain-permutation  $A\ B = \{(f, f') \in (A \rightarrow_E B) \times (A \rightarrow_E B). \exists p. p\ \textit{permutes}
 $A \wedge (\forall x \in A. f\ x = f'\ (p\ x))\}$$ 
```

```

lemma equiv-domain-permutation:

```

```

  equiv  $(A \rightarrow_E B)$  (domain-permutation  $A\ B$ )

```

```

proof (rule equivI)

```

```

  show refl-on  $(A \rightarrow_E B)$  (domain-permutation  $A\ B$ )

```

```

  proof (rule refl-onI)

```

```

    show domain-permutation  $A\ B \subseteq (A \rightarrow_E B) \times (A \rightarrow_E B)$ 

```

```

      unfolding domain-permutation-def by auto

```

```

  next

```

```

    fix  $f$ 

```

```

    assume  $f \in A \rightarrow_E B$ 

```

```

    from this show  $(f, f) \in \textit{domain-permutation}\ A\ B$ 

```

```

    using permutes-id unfolding domain-permutation-def by fastforce
  qed
next
show sym (domain-permutation A B)
proof (rule symI)
  fix f f'
  assume  $(f, f') \in \text{domain-permutation } A B$ 
  from this obtain p where p permutes A and  $\forall x \in A. f x = f' (p x)$ 
    unfolding domain-permutation-def by auto
  from  $\langle (f, f') \in \text{domain-permutation } A B \rangle$  have  $f \in A \rightarrow_E B$   $f' \in A \rightarrow_E B$ 
    unfolding domain-permutation-def by auto
  moreover from  $\langle p \text{ permutes } A \rangle$  have inv p permutes A
    by (simp add: permutes-inv)
  moreover from  $\langle p \text{ permutes } A \rangle$   $\langle \forall x \in A. f x = f' (p x) \rangle$  have  $\forall x \in A. f' x = f$ 
    (inv p x)
    using permutes-in-image permutes-inverses(1) by (metis (mono-tags, opaque-lifting))
  ultimately show  $(f', f) \in \text{domain-permutation } A B$ 
    unfolding domain-permutation-def by auto
  qed
next
show trans (domain-permutation A B)
proof (rule transI)
  fix f f' f''
  assume  $(f, f') \in \text{domain-permutation } A B$   $(f', f'') \in \text{domain-permutation } A B$ 
  from  $\langle (f, f') \in \rightarrow \rangle$  obtain p where p permutes A and  $\forall x \in A. f x = f' (p x)$ 
    unfolding domain-permutation-def by auto
  from  $\langle (f', f'') \in \rightarrow \rangle$  obtain p' where p' permutes A and  $\forall x \in A. f' x = f'' (p' x)$ 
    (x)
    unfolding domain-permutation-def by auto
  from  $\langle (f, f') \in \text{domain-permutation } A B \rangle$  have  $f \in A \rightarrow_E B$ 
    unfolding domain-permutation-def by auto
  moreover from  $\langle (f', f'') \in \text{domain-permutation } A B \rangle$  have  $f'' \in A \rightarrow_E B$ 
    unfolding domain-permutation-def by auto
  moreover from  $\langle p \text{ permutes } A \rangle$   $\langle p' \text{ permutes } A \rangle$  have  $(p' \circ p) \text{ permutes } A$ 
    by (simp add: permutes-compose)
  moreover have  $\forall x \in A. f x = f'' ((p' \circ p) x)$ 
    using  $\langle \forall x \in A. f x = f' (p x) \rangle$   $\langle \forall x \in A. f' x = f'' (p' x) \rangle$   $\langle p \text{ permutes } A \rangle$ 
    by (simp add: permutes-in-image)
  ultimately show  $(f, f'') \in \text{domain-permutation } A B$ 
    unfolding domain-permutation-def by auto
  qed
qed

```

3.1.1 Respecting Functions

lemma *inj-on-respects-domain-permutation*:

$(\lambda f. \text{inj-on } f A)$ *respects domain-permutation* *A B*

proof (*rule congruentI*)

fix *f f'*

```

assume  $(f, f') \in \text{domain-permutation } A B$ 
from this obtain  $p$  where  $p: p \text{ permutes } A \ \forall x \in A. f x = f' (p x)$ 
  unfolding domain-permutation-def by auto
have  $\text{inv-}p: \forall x \in A. f' x = f (\text{inv } p x)$ 
  using  $p$  by  $(\text{metis permutes-inverses}(1) \text{ permutes-not-in})$ 
show  $\text{inj-on } f A \longleftrightarrow \text{inj-on } f' A$ 
proof
  assume  $\text{inj-on } f A$ 
  show  $\text{inj-on } f' A$ 
  proof  $(\text{rule inj-onI})$ 
    fix  $a a'$ 
    assume  $a \in A \ a' \in A \ f' a = f' a'$ 
    from this  $\langle p \text{ permutes } A \rangle$  have  $\text{inv } p a \in A \ \text{inv } p a' \in A$ 
      by  $(\text{simp add: permutes-in-image permutes-inv})+$ 
    have  $f (\text{inv } p a) = f (\text{inv } p a')$ 
      using  $\langle f' a = f' a' \rangle \langle a \in A \rangle \langle a' \in A \rangle \text{inv-}p$  by auto
    from  $\langle \text{inj-on } f A \rangle$  this  $\langle \text{inv } p a \in A \rangle \langle \text{inv } p a' \in A \rangle$  have  $\text{inv } p a = \text{inv } p a'$ 
      using inj-on-contrad by fastforce
    from this show  $a = a'$ 
      by  $(\text{metis } \langle p \text{ permutes } A \rangle \text{ permutes-inverses}(1))$ 
  qed
next
  assume  $\text{inj-on } f' A$ 
  from this  $p$  show  $\text{inj-on } f A$ 
    unfolding inj-on-def
    by  $(\text{metis inj-on-contrad permutes-in-image permutes-inj-on})$ 
  qed
qed

```

lemma *image-respects-domain-permutation:*

```

 $(\lambda f. f' A) \text{ respects } (\text{domain-permutation } A B)$ 
proof  $(\text{rule congruentI})$ 
  fix  $f f'$ 
  assume  $(f, f') \in \text{domain-permutation } A B$ 
  from this obtain  $p$  where  $p: p \text{ permutes } A$  and  $f\text{-eq}: \forall x \in A. f x = f' (p x)$ 
    unfolding domain-permutation-def by auto
  show  $f' A = f' A$ 
  proof
    from  $p \text{ f-eq}$  show  $f' A \subseteq f' A$ 
      by  $(\text{auto simp add: permutes-in-image})$ 
    next
      from  $\langle p \text{ permutes } A \rangle \langle \forall x \in A. f x = f' (p x) \rangle$  have  $\forall x \in A. f' x = f (\text{inv } p x)$ 
        using permutes-in-image permutes-inverses(1) by  $(\text{metis } (\text{mono-tags, opaque-lifting}))$ 
      from this show  $f' A \subseteq f' A$ 
        using  $\langle p \text{ permutes } A \rangle$  by  $(\text{auto simp add: permutes-inv permutes-in-image})$ 
    qed
  qed

```

lemma *surjective-respects-domain-permutation:*

($\lambda f. f \text{ ' } A = B$) respects domain-permutation $A B$
by (*metis image-respects-domain-permutation congruentD congruentI*)

lemma *bij-betw-respects-domain-permutation:*

($\lambda f. \text{bij-betw } f A B$) respects domain-permutation $A B$

proof (*rule congruentI*)

fix $f f'$

assume $(f, f') \in \text{domain-permutation } A B$

from this obtain p **where** p permutes A **and** $\forall x \in A. f x = f' (p x)$

unfolding *domain-permutation-def* **by** *auto*

have $\text{bij-betw } f A B \longleftrightarrow \text{bij-betw } (f' \circ p) A B$

using $\langle \forall x \in A. f x = f' (p x) \rangle$

by (*metis (mono-tags, opaque-lifting) bij-betw-cong comp-apply*)

also have $\dots \longleftrightarrow \text{bij-betw } f' A B$

using $\langle p \text{ permutes } A \rangle$

by (*auto intro!: bij-betw-comp-iff[symmetric] permutes-imp-bij*)

finally show $\text{bij-betw } f A B \longleftrightarrow \text{bij-betw } f' A B$.

qed

lemma *image-mset-respects-domain-permutation:*

shows ($\lambda f. \text{image-mset } f (\text{mset-set } A)$) respects (*domain-permutation* $A B$)

proof (*rule congruentI*)

fix $f f'$

assume $(f, f') \in \text{domain-permutation } A B$

from this obtain p **where** p permutes A **and** $\forall x \in A. f x = f' (p x)$

unfolding *domain-permutation-def* **by** *auto*

from this show $\text{image-mset } f (\text{mset-set } A) = \text{image-mset } f' (\text{mset-set } A)$

using *permutes-implies-image-mset-eq* **by** *fastforce*

qed

3.2 Permutation on the Range

definition *range-permutation*

where

$\text{range-permutation } A B = \{(f, f') \in (A \rightarrow_E B) \times (A \rightarrow_E B). \exists p. p \text{ permutes } B \wedge (\forall x \in A. f x = p (f' x))\}$

lemma *equiv-range-permutation:*

equiv $(A \rightarrow_E B)$ (*range-permutation* $A B$)

proof (*rule equivI*)

show *refl-on* $(A \rightarrow_E B)$ (*range-permutation* $A B$)

proof (*rule refl-onI*)

show *range-permutation* $A B \subseteq (A \rightarrow_E B) \times (A \rightarrow_E B)$

unfolding *range-permutation-def* **by** *auto*

next

fix f

assume $f \in A \rightarrow_E B$

from this show $(f, f) \in \text{range-permutation } A B$

using *permutes-id* **unfolding** *range-permutation-def* **by** *fastforce*


```

qed
next
show sym (range-permutation A B)
proof (rule symI)
  fix f f'
  assume (f, f') ∈ range-permutation A B
  from this obtain p where p permutes B and  $\forall x \in A. f x = p (f' x)$ 
  unfolding range-permutation-def by auto
  from  $\langle (f, f') \in \text{range-permutation } A \ B \rangle$  have  $f \in A \rightarrow_E B$   $f' \in A \rightarrow_E B$ 
  unfolding range-permutation-def by auto
  moreover from  $\langle p \text{ permutes } B \rangle$  have inv p permutes B
  by (simp add: permutes-inv)
  moreover from  $\langle p \text{ permutes } B \rangle \langle \forall x \in A. f x = p (f' x) \rangle$  have  $\forall x \in A. f' x =$ 
  inv p (f x)
  by (simp add: permutes-inverses(2))
  ultimately show (f', f) ∈ range-permutation A B
  unfolding range-permutation-def by auto
qed
next
show trans (range-permutation A B)
proof (rule transI)
  fix f f' f''
  assume (f, f') ∈ range-permutation A B (f', f'') ∈ range-permutation A B
  from  $\langle (f, f') \in \rightarrow \rangle$  obtain p where p permutes B and  $\forall x \in A. f x = p (f' x)$ 
  unfolding range-permutation-def by auto
  from  $\langle (f', f'') \in \rightarrow \rangle$  obtain p' where p' permutes B and  $\forall x \in A. f' x = p' (f''$ 
  x)
  unfolding range-permutation-def by auto
  from  $\langle (f, f') \in \text{range-permutation } A \ B \rangle$  have  $f \in A \rightarrow_E B$ 
  unfolding range-permutation-def by auto
  moreover from  $\langle (f', f'') \in \text{range-permutation } A \ B \rangle$  have  $f'' \in A \rightarrow_E B$ 
  unfolding range-permutation-def by auto
  moreover from  $\langle p \text{ permutes } B \rangle \langle p' \text{ permutes } B \rangle$  have  $(p \circ p') \text{ permutes } B$ 
  by (simp add: permutes-compose)
  moreover have  $\forall x \in A. f x = (p \circ p') (f'' x)$ 
  using  $\langle \forall x \in A. f x = p (f' x) \rangle \langle \forall x \in A. f' x = p' (f'' x) \rangle$  by auto
  ultimately show (f, f'') ∈ range-permutation A B
  unfolding range-permutation-def by auto
qed
qed

```

3.2.1 Respecting Functions

lemma *inj-on-respects-range-permutation:*

$(\lambda f. \text{inj-on } f \ A) \text{ respects range-permutation } A \ B$

proof (rule congruentI)

fix f f'

assume (f, f') ∈ range-permutation A B

from this obtain p where p: p permutes B $\forall x \in A. f x = p (f' x)$

```

  unfolding range-permutation-def by auto
  have inv-p:  $\forall x \in A. f' x = \text{inv } p (f x)$ 
  using p by (simp add: permutes-inverses(2))
  show inj-on f A  $\longleftrightarrow$  inj-on f' A
  proof
    assume inj-on f A
    from this p show inj-on f' A
      unfolding inj-on-def by auto
  next
    assume inj-on f' A
    from this inv-p show inj-on f A
      unfolding inj-on-def by auto
  qed
qed

```

lemma *surj-on-respects-range-permutation:*

```

( $\lambda f. f \text{ ' } A = B$ ) respects range-permutation A B
proof (rule congruentI)
  fix f f'
  assume a:  $(f, f') \in \text{range-permutation } A B$ 
  from this have  $f \in A \rightarrow_E B$   $f' \in A \rightarrow_E B$ 
  unfolding range-permutation-def by auto
  from a obtain p where p: p permutes B  $\forall x \in A. f x = p (f' x)$ 
  unfolding range-permutation-def by auto
  have 1:  $f \text{ ' } A = (\lambda x. p (f' x)) \text{ ' } A$ 
  using p by (meson image-cong)
  have 2:  $\text{inv } p \text{ ' } ((\lambda x. p (f' x)) \text{ ' } A) = f' \text{ ' } A$ 
  using p by (simp add: image-image image-inv-f-f permutes-inj)
  show  $(f \text{ ' } A = B) = (f' \text{ ' } A = B)$ 
  proof
    assume  $f \text{ ' } A = B$ 
    from this 1 2 show  $f' \text{ ' } A = B$ 
      using p by (simp add: permutes-image permutes-inv)
  next
    assume  $f' \text{ ' } A = B$ 
    from this 1 2 show  $f \text{ ' } A = B$ 
      using p by (metis image-image permutes-image)
  qed
qed

```

lemma *bij-betw-respects-range-permutation:*

```

( $\lambda f. \text{bij-betw } f A B$ ) respects range-permutation A B
proof (rule congruentI)
  fix f f'
  assume  $(f, f') \in \text{range-permutation } A B$ 
  from this obtain p where p permutes B and  $\forall x \in A. f x = p (f' x)$ 
  and  $f' \in A \rightarrow_E B$ 
  unfolding range-permutation-def by auto
  have  $\text{bij-betw } f A B \longleftrightarrow \text{bij-betw } (p \circ f') A B$ 

```

using $\langle \forall x \in A. f x = p (f' x) \rangle$
by (*metis* (*mono-tags*, *opaque-lifting*) *bij-betw-cong comp-apply*)
also have ... \longleftrightarrow *bij-betw* $f' A B$
using $\langle f' \in A \rightarrow_E B \rangle$ $\langle p$ *permutes* $B \rangle$
by (*auto intro!*: *bij-betw-comp-iff2*[*symmetric*] *permutes-imp-bij*)
finally show *bij-betw* $f A B \longleftrightarrow$ *bij-betw* $f' A B$.
qed

lemma *domain-partitions-respects-range-permutation*:

$(\lambda f. (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\})$ *respects range-permutation* $A B$
proof (*rule congruentI*)

fix $f f'$
assume $(f, f') \in$ *range-permutation* $A B$
from *this* **obtain** p **where** p : p *permutes* $B \forall x \in A. f x = p (f' x)$
unfolding *range-permutation-def* **by** *blast*
have $\{\} \in (\lambda b. \{x \in A. f' x = b\}) ' B \longleftrightarrow \neg (\forall b \in B. \exists x \in A. f' x = b)$ **by**
auto
also have $(\forall b \in B. \exists x \in A. f' x = b) \longleftrightarrow (\forall b \in B. \exists x \in A. p (f' x) = b)$
proof
assume $\forall b \in B. \exists x \in A. f' x = b$
from *this* **show** $\forall b \in B. \exists x \in A. p (f' x) = b$
using $\langle p$ *permutes* $B \rangle$ **unfolding** *permutes-def* **by** *metis*
next
assume $\forall b \in B. \exists x \in A. p (f' x) = b$
from *this* **show** $\forall b \in B. \exists x \in A. f' x = b$
using $\langle p$ *permutes* $B \rangle$ **by** (*metis* *bij-betwE permutes-imp-bij permutes-inverses*(2))
qed
also have $\neg (\forall b \in B. \exists x \in A. p (f' x) = b) \longleftrightarrow \{\} \in (\lambda b. \{x \in A. p (f' x) = b\})$
 $' B$ **by** *auto*
finally have $\{\} \in (\lambda b. \{x \in A. f' x = b\}) ' B \longleftrightarrow \{\} \in (\lambda b. \{x \in A. p (f' x) = b\}) ' B$.
moreover have $(\lambda b. \{x \in A. f' x = b\}) ' B = (\lambda b. \{x \in A. p (f' x) = b\}) ' B$
using $\langle p$ *permutes* $B \rangle$ *permutes-implies-inv-image-on-eq* **by** *blast*
ultimately have $(\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\} = (\lambda b. \{x \in A. p (f' x) = b\}) ' B - \{\{\}\}$ **by** *auto*
also have ... $= (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}$
using $\langle \forall x \in A. f x = p (f' x) \rangle$ *Collect-cong image-cong* **by** *auto*
finally show $(\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\}$..
qed

3.3 Permutation on the Domain and the Range

definition *domain-and-range-permutation*

where

$domain-and-range-permutation A B = \{(f, f') \in (A \rightarrow_E B) \times (A \rightarrow_E B).$
 $\exists p_A p_B. p_A$ *permutes* $A \wedge p_B$ *permutes* $B \wedge (\forall x \in A. f x = p_B (f' (p_A x)))\}$

lemma *equiv-domain-and-range-permutation*:

```

equiv (A →E B) (domain-and-range-permutation A B)
proof (rule equivI)
show refl-on (A →E B) (domain-and-range-permutation A B)
proof (rule refl-onI)
  show domain-and-range-permutation A B ⊆ (A →E B) × (A →E B)
    unfolding domain-and-range-permutation-def by auto
next
fix f
assume f ∈ A →E B
from this show (f, f) ∈ domain-and-range-permutation A B
  using permutes-id[of A] permutes-id[of B]
  unfolding domain-and-range-permutation-def by fastforce
qed
next
show sym (domain-and-range-permutation A B)
proof (rule symI)
  fix f f'
  assume (f, f') ∈ domain-and-range-permutation A B
  from this obtain pA pB where pA permutes A pB permutes B and ∀ x ∈ A. f
x = pB (f' (pA x))
  unfolding domain-and-range-permutation-def by auto
  from ⟨(f, f') ∈ domain-and-range-permutation A B⟩ have f: f ∈ A →E B f'
∈ A →E B
  unfolding domain-and-range-permutation-def by auto
  moreover from ⟨pA permutes A⟩ ⟨pB permutes B⟩ have inv pA permutes A
inv pB permutes B
  by (auto simp add: permutes-inv)
  moreover from ⟨∀ x ∈ A. f x = pB (f' (pA x))⟩ have ∀ x ∈ A. f' x = inv pB (f
(inv pA x))
  using ⟨pA permutes A⟩ ⟨pB permutes B⟩ ⟨inv pA permutes A⟩ ⟨inv pB permutes
B⟩
  by (metis (no-types, lifting) bij-betwE bij-inv-eq-iff permutes-bij permutes-imp-bij)
  ultimately show (f', f) ∈ domain-and-range-permutation A B
  unfolding domain-and-range-permutation-def by auto
qed
next
show trans (domain-and-range-permutation A B)
proof (rule transI)
  fix f f' f''
  assume (f, f') ∈ domain-and-range-permutation A B
  assume (f', f'') ∈ domain-and-range-permutation A B
  from ⟨(f, f') ∈ →⟩ obtain pA pB where
pA permutes A pB permutes B and ∀ x ∈ A. f x = pB (f' (pA x))
  unfolding domain-and-range-permutation-def by auto
  from ⟨(f', f'') ∈ →⟩ obtain p'A p'B where
p'A permutes A p'B permutes B and ∀ x ∈ A. f' x = p'B (f'' (p'A x))
  unfolding domain-and-range-permutation-def by auto
  from ⟨(f, f') ∈ domain-and-range-permutation A B⟩ have f ∈ A →E B
  unfolding domain-and-range-permutation-def by auto

```

moreover from $\langle f', f'' \rangle \in \text{domain-and-range-permutation } A \ B \rangle$ **have** $f'' \in A \rightarrow_E B$
unfolding *domain-and-range-permutation-def* **by** *auto*
moreover from $\langle p_A \text{ permutes } A \rangle \langle p'_A \text{ permutes } A \rangle$ **have** $\langle p'_A \circ p_A \text{ permutes } A \rangle$
by (*simp add: permutes-compose*)
moreover from $\langle p_B \text{ permutes } B \rangle \langle p'_B \text{ permutes } B \rangle$ **have** $\langle p_B \circ p'_B \text{ permutes } B \rangle$
by (*simp add: permutes-compose*)
moreover have $\forall x \in A. f x = (p_B \circ p'_B) (f'' ((p'_A \circ p_A) x))$
using $\langle \forall x \in A. f' x = p'_B (f'' (p'_A x)) \rangle \langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle \langle p_A \text{ permutes } A \rangle$
by (*simp add: permutes-in-image*)
ultimately show $\langle f, f'' \rangle \in \text{domain-and-range-permutation } A \ B$
unfolding *domain-and-range-permutation-def* **by** *fastforce*
qed
qed

3.3.1 Respecting Functions

lemma *inj-on-respects-domain-and-range-permutation:*

$(\lambda f. \text{inj-on } f \ A) \text{ respects domain-and-range-permutation } A \ B$

proof (*rule congruentI*)

fix $f \ f'$

assume $\langle f, f' \rangle \in \text{domain-and-range-permutation } A \ B$

from this obtain $p_A \ p_B$ **where** $p_A \text{ permutes } A \ p_B \text{ permutes } B$ **and** $\forall x \in A. f x = p_B (f' (p_A x))$

unfolding *domain-and-range-permutation-def* **by** *auto*

from $\langle f, f' \rangle \in \text{domain-and-range-permutation } A \ B$ **have** $f' \text{ ' } A \subseteq B$

unfolding *domain-and-range-permutation-def* **by** *auto*

from $\langle p_A \text{ permutes } A \rangle$ **have** $p_A \text{ ' } A = A$ **by** (*auto simp add: permutes-image*)

from $\langle p_A \text{ permutes } A \rangle$ **have** $\text{inj-on } p_A \ A$

using *bij-betw-imp-inj-on permutes-imp-bij* **by** *blast*

from $\langle p_B \text{ permutes } B \rangle$ **have** $\text{inj-on } p_B \ B$

using *bij-betw-imp-inj-on permutes-imp-bij* **by** *blast*

show $\text{inj-on } f \ A \longleftrightarrow \text{inj-on } f' \ A$

proof –

have $\text{inj-on } f \ A \longleftrightarrow \text{inj-on } (\lambda x. p_B (f' (p_A x))) \ A$

using $\langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle$ *inj-on-cong comp-apply* **by** *fastforce*

have $\text{inj-on } f \ A \longleftrightarrow \text{inj-on } (p_B \circ f' \circ p_A) \ A$

by (*simp add: $\langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle$ inj-on-def*)

also have $\text{inj-on } (p_B \circ f' \circ p_A) \ A \longleftrightarrow \text{inj-on } (p_B \circ f') \ A$

using $\langle \text{inj-on } p_A \ A \rangle \langle p_A \text{ ' } A = A \rangle$

by (*auto dest: inj-on-imageI intro: comp-inj-on*)

also have $\text{inj-on } (p_B \circ f') \ A \longleftrightarrow \text{inj-on } f' \ A$

using $\langle \text{inj-on } p_B \ B \rangle \langle f' \text{ ' } A \subseteq B \rangle$

by (*auto dest: inj-on-imageI2 intro: comp-inj-on subset-inj-on*)

finally show *?thesis* .

qed

qed

lemma *surjective-respects-domain-and-range-permutation:*

$(\lambda f. f \text{ ' } A = B)$ respects domain-and-range-permutation $A B$

proof (rule congruentI)

fix $f f'$

assume $(f, f') \in \text{domain-and-range-permutation } A B$

from this obtain $p_A p_B$ where

$\text{permutes } p_A \text{ permutes } A p_B \text{ permutes } B$ and $\forall x \in A. f x = p_B (f' (p_A x))$

unfolding domain-and-range-permutation-def by auto

from $\text{permutes } p_A \text{ ' } A = A p_B \text{ ' } B = B$ by (auto simp add: permutes-image)

from $\langle p_B \text{ permutes } B \rangle$ have $\text{inj } p_B$ by (simp add: permutes-inj)

show $(f \text{ ' } A = B) \longleftrightarrow (f' \text{ ' } A = B)$

proof –

have $f \text{ ' } A = B \longleftrightarrow (\lambda x. p_B (f' (p_A x))) \text{ ' } A = B$

using $\langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle$ by (metis (mono-tags, lifting) image-cong)

also have $(\lambda x. p_B (f' (p_A x))) \text{ ' } A = B \longleftrightarrow (\lambda x. p_B (f' x)) \text{ ' } A = B$

using $\langle p_A \text{ ' } A = A \rangle$ by (metis image-image)

also have $(\lambda x. p_B (f' x)) \text{ ' } A = B \longleftrightarrow (f' \text{ ' } A = B)$

using $\langle p_B \text{ ' } B = B \rangle \langle \text{inj } p_B \rangle$ by (metis image-image image-inv-f-f)

finally show ?thesis .

qed

qed

lemma *bij-betw-respects-domain-and-range-permutation:*

$(\lambda f. \text{bij-betw } f A B)$ respects domain-and-range-permutation $A B$

proof (rule congruentI)

fix $f f'$

assume $(f, f') \in \text{domain-and-range-permutation } A B$

from this obtain $p_A p_B$ where $p_A \text{ permutes } A p_B \text{ permutes } B$

and $\forall x \in A. f x = p_B (f' (p_A x))$ and $f' \in A \rightarrow_E B$

unfolding domain-and-range-permutation-def by auto

have $\text{bij-betw } f A B \longleftrightarrow \text{bij-betw } (p_B \circ f' \circ p_A) A B$

using $\langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle \text{bij-betw-congI}$ by fastforce

also have $\dots \longleftrightarrow \text{bij-betw } (p_B \circ f') A B$

using $\langle p_A \text{ permutes } A \rangle$

by (auto intro!: bij-betw-comp-iff[symmetric] permutes-imp-bij)

also have $\dots \longleftrightarrow \text{bij-betw } f' A B$

using $\langle f' \in A \rightarrow_E B \rangle \langle p_B \text{ permutes } B \rangle$

by (auto intro!: bij-betw-comp-iff2[symmetric] permutes-imp-bij)

finally show $\text{bij-betw } f A B \longleftrightarrow \text{bij-betw } f' A B$.

qed

lemma *count-image-mset':*

$\text{count } (\text{image-mset } f A) x = \text{sum } (\text{count } A) \{x' \in \text{set-mset } A. f x' = x\}$

proof –

have $\text{count } (\text{image-mset } f A) x = \text{sum } (\text{count } A) (f \text{ - ' } \{x\} \cap \text{set-mset } A)$

unfolding count-image-mset ..

also have $\dots = \text{sum } (\text{count } A) \{x' \in \text{set-mset } A. f x' = x\}$

proof –
have $(f - \{x\} \cap \text{set-mset } A) = \{x' \in \text{set-mset } A. f x' = x\}$ **by** *blast*
from this show *?thesis by simp*
qed
finally show *?thesis* .
qed

lemma *multiset-of-partition-cards-respects-domain-and-range-permutation:*

assumes *finite B*
shows $(\lambda f. \text{image-mset } (\lambda X. \text{card } X) (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\})))$ *respects domain-and-range-permutation A B*
proof (*rule congruentI*)
fix $f f'$
assume $(f, f') \in \text{domain-and-range-permutation } A B$
from this obtain $p_A p_B$ **where** p_A *permutes A* p_B *permutes B* $\forall x \in A. f x = p_B (f' (p_A x))$
unfolding *domain-and-range-permutation-def* **by** *auto*
have $(\lambda b. \{x \in A. f x = b\}) ' B = (\lambda b. \{x \in A. p_B (f' (p_A x)) = b\}) ' B$
using $\langle \forall x \in A. f x = p_B (f' (p_A x)) \rangle$ **by** *auto*
from this have $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}))$
 $=$
 $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. p_B (f' (p_A x)) = b\}) ' B - \{\{\}\}))$ **by**
simp
also have $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. p_B (f' (p_A x)) = b\}) ' B - \{\{\}\})) =$
 $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' (p_A x) = b\}) ' B - \{\{\}\}))$
using *permutes-implies-inv-image-on-eq[OF $\langle p_B \text{ permutes } B \rangle$, of A]* **by** *metis*
also have $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' (p_A x) = b\}) ' B - \{\{\}\}))$
 $=$
 $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\}))$
proof (*rule multiset-eqI*)
fix n
have $\text{bij-betw } (\lambda X. p_A ' X) \{X \in (\lambda b. \{x \in A. f' (p_A x) = b\}) ' B - \{\{\}\}. \text{card } X = n\} \{X \in (\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\}. \text{card } X = n\}$
proof (*rule bij-betw-byWitness*)
show $\forall X \in \{X \in (\lambda b. \{x \in A. f' (p_A x) = b\}) ' B - \{\{\}\}. \text{card } X = n\}. \text{inv } p_A ' p_A ' X = X$
by (*meson $\langle p_A \text{ permutes } A \rangle \text{image-inv-f-f permutes-inj}$*)
show $\forall X \in \{X \in (\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\}. \text{card } X = n\}. p_A ' \text{inv } p_A ' X = X$
by (*meson $\langle p_A \text{ permutes } A \rangle \text{image-f-inv-f permutes-surj}$*)
show $(\lambda X. p_A ' X) ' \{X \in (\lambda b. \{x \in A. f' (p_A x) = b\}) ' B - \{\{\}\}. \text{card } X = n\} \subseteq \{X \in (\lambda b. \{x \in A. f' x = b\}) ' B - \{\{\}\}. \text{card } X = n\}$
proof –
have $\text{card } (p_A ' \{x \in A. f' (p_A x) = b\}) = \text{card } \{x \in A. f' (p_A x) = b\}$ **for**
 b
proof –
have $\text{inj-on } p_A \{x \in A. f' (p_A x) = b\}$
by (*metis (no-types, lifting) $\langle p_A \text{ permutes } A \rangle \text{injD inj-onI permutes-inj}$*)

```

    from this show ?thesis by (simp add: card-image)
  qed
  moreover have  $p_A \text{ ' } \{x \in A. f' (p_A x) = b\} = \{x \in A. f' x = b\}$  for  $b$ 
  proof
    show  $p_A \text{ ' } \{x \in A. f' (p_A x) = b\} \subseteq \{x \in A. f' x = b\}$ 
      by (auto simp add:  $\langle p_A \text{ permutes } A \rangle$  permutes-in-image)
    show  $\{x \in A. f' x = b\} \subseteq p_A \text{ ' } \{x \in A. f' (p_A x) = b\}$ 
      proof
        fix  $x$ 
        assume  $x \in \{x \in A. f' x = b\}$ 
        moreover have  $p_A (inv p_A x) = x$ 
          using  $\langle p_A \text{ permutes } A \rangle$  permutes-inverses(1) by fastforce
        moreover from  $\langle x \in \{x \in A. f' x = b\} \rangle$  have  $inv p_A x \in A$ 
          by (simp add:  $\langle p_A \text{ permutes } A \rangle$  permutes-in-image permutes-inv)
        ultimately show  $x \in p_A \text{ ' } \{x \in A. f' (p_A x) = b\}$ 
          by (auto intro: image-eqI[where  $x=inv p_A x$ ])
      proof
    qed
  qed
  ultimately show ?thesis by auto
  qed
  show  $(\lambda X. inv p_A \text{ ' } X) \text{ ' } \{X \in (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}. card X = n\} \subseteq \{X \in (\lambda b. \{x \in A. f' (p_A x) = b\}) \text{ ' } B - \{\{\}\}. card X = n\}$ 
  proof -
    have  $card (inv p_A \text{ ' } \{x \in A. f' x = b\}) = card \{x \in A. f' x = b\}$  for  $b$ 
    proof -
      have  $inj\text{-on } (inv p_A) \{x \in A. f' x = b\}$ 
        by (metis (no-types, lifting)  $\langle p_A \text{ permutes } A \rangle$  injD inj-onI permutes-surj
        surj-imp-inj-inv)
      from this show ?thesis by (simp add: card-image)
    qed
  moreover have  $inv p_A \text{ ' } \{x \in A. f' x = b\} = \{x \in A. f' (p_A x) = b\}$  for  $b$ 
  proof
    show  $inv p_A \text{ ' } \{x \in A. f' x = b\} \subseteq \{x \in A. f' (p_A x) = b\}$ 
      using  $\langle p_A \text{ permutes } A \rangle$ 
      by (auto simp add: permutes-in-image permutes-inv permutes-inverses(1))
    show  $\{x \in A. f' (p_A x) = b\} \subseteq inv p_A \text{ ' } \{x \in A. f' x = b\}$ 
      proof
        fix  $x$ 
        assume  $x \in \{x \in A. f' (p_A x) = b\}$ 
        moreover have  $inv p_A (p_A x) = x$ 
          by (meson  $\langle p_A \text{ permutes } A \rangle$  permutes-inverses(2))
        moreover from  $\langle x \in \{x \in A. f' (p_A x) = b\} \rangle$  have  $p_A x \in A$ 
          by (simp add:  $\langle p_A \text{ permutes } A \rangle$  permutes-in-image)
        ultimately show  $x \in inv p_A \text{ ' } \{x \in A. f' x = b\}$ 
          by (auto intro: image-eqI[where  $x=p_A x$ ])
      proof
    qed
  qed
  ultimately show ?thesis by auto
  qed

```


qed
from *this* **have** $\text{card } \{x' \in (\lambda b. \{x \in A. f' (p_A x) = b\}) \text{ ' } B - \{\{\}\}\}. \text{card } x' = n\} = \text{card } \{x' \in (\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}\}. \text{card } x' = n\}$
by (*rule bij-betw-same-card*)
from *this* **show** $\text{count } (\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' (p_A x) = b\}) \text{ ' } B - \{\{\}\}\})) n = \text{count } (\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}\})) n$
using $\langle \text{finite } B \rangle$ **by** (*simp add: count-image-mset'*)
qed
finally show $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}\})) = \text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f' x = b\}) \text{ ' } B - \{\{\}\}\})) .$
qed
end

4 Functions from A to B

theory *Twelvefold-Way-Entry1*
imports *Preliminaries*
begin

Note that the cardinality theorems of both structures, lists and finite functions, are already available. Hence, this development creates the bijection between those two structures and transfers the one cardinality theorem to the other structures and vice versa, although not strictly needed as both cardinality theorems were already available.

4.1 Definition of Bijections

definition *sequence-of* $:: 'a \text{ set} \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'b \text{ list}$
where
 $\text{sequence-of } A \text{ enum } f = \text{map } (\lambda n. f (\text{enum } n)) [0..<\text{card } A]$

definition *function-of* $:: 'a \text{ set} \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow 'b \text{ list} \Rightarrow ('a \Rightarrow 'b)$
where
 $\text{function-of } A \text{ enum } xs = (\lambda a. \text{if } a \in A \text{ then } xs ! \text{inv-into } \{0..<\text{length } xs\} \text{ enum } a \text{ else undefined})$

4.2 Properties for Bijections

lemma *nth-sequence-of*:
assumes $i < \text{card } A$
shows $(\text{sequence-of } A \text{ enum } f) ! i = f (\text{enum } i)$
using *assms unfolding sequence-of-def* **by** *auto*

lemma *nth-sequence-of-inv-into*:
assumes *bij-betw* $\text{enum } \{0..<\text{card } A\} A$
assumes $a \in A$

shows $(\text{sequence-of } A \text{ enum } f) ! (\text{inv-into } \{0..<\text{card } A\} \text{ enum } a) = f a$
proof –
have $\text{inv-into } \{0..<\text{card } A\} \text{ enum } a \in \{0..<\text{card } A\}$
using *assms bij-betwE bij-betw-inv-into* **by** *blast*
from *this assms* **show** $(\text{sequence-of } A \text{ enum } f) ! (\text{inv-into } \{0..<\text{card } A\} \text{ enum } a)$
 $= f a$
unfolding *sequence-of-def* **by** *(simp add: bij-betw-inv-into-right)*
qed

lemma *set-sequence-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\} A$
assumes $f \in A \rightarrow_E B$
shows $\text{set } (\text{sequence-of } A \text{ enum } f) \subseteq B$
using *PiE bij-betwE assms*
unfolding *sequence-of-def* **by** *fastforce*

lemma *length-sequence-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\} A$
assumes $f \in A \rightarrow_E B$
shows $\text{length } (\text{sequence-of } A \text{ enum } f) = \text{card } A$
using *assms* **unfolding** *sequence-of-def* **by** *simp*

lemma *function-of-enum*:
assumes *bij-betw enum* $\{0..<\text{card } A\} A$
assumes $\text{length } xs = \text{card } A$
assumes $i < \text{card } A$
shows $\text{function-of } A \text{ enum } xs (\text{enum } i) = xs ! i$
using *assms* **unfolding** *function-of-def*
by *(auto simp add: bij-betw-inv-into-left bij-betwE)*

lemma *function-of-in-extensional-funcset*:
assumes *bij-betw enum* $\{0..<\text{card } A\} A$
assumes $\text{set } xs \subseteq B \text{ length } xs = \text{card } A$
shows $\text{function-of } A \text{ enum } xs \in A \rightarrow_E B$
proof
fix x
assume $x \in A$
have $\text{inv-into } \{0..<\text{length } xs\} \text{ enum } x \in \{0..<\text{length } xs\}$
using $\langle x \in A \rangle$ *assms(1, 3)* **by** *(metis bij-betw-def inv-into-into)*
from *this* **have** $xs ! \text{inv-into } \{0..<\text{length } xs\} \text{ enum } x \in \text{set } xs$ **by** *simp*
from *this* $\langle \text{set } xs \subseteq B \rangle$ **show** $\text{function-of } A \text{ enum } xs \ x \in B$
using $\langle x \in A \rangle$ **unfolding** *function-of-def* **by** *auto*
next
fix x
assume $x \notin A$
from *this* **show** $\text{function-of } A \text{ enum } xs \ x = \text{undefined}$
unfolding *function-of-def* **by** *simp*
qed

lemma *sequence-of-function-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\}$ A
assumes *set* $xs \subseteq B$ $\text{length } xs = \text{card } A$
shows *sequence-of* A *enum* (*function-of* A *enum* xs) = xs
proof (*rule nth-equalityI*)
have *function-of* A *enum* $xs \in A \rightarrow_E B$
using *assms* **by** (*rule function-of-in-extensional-funcset*)
from *this* **show** $\text{length} (\text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs)) = \text{length } xs$
using *assms*(1,3) **by** (*simp add: length-sequence-of*)
from *this* **show** $\bigwedge i. i < \text{length} (\text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs))$
 $\implies \text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs) ! i = xs ! i$
using *assms* **by** (*auto simp add: nth-sequence-of function-of-enum*)
qed

lemma *function-of-sequence-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\}$ A
assumes $f \in A \rightarrow_E B$
shows *function-of* A *enum* (*sequence-of* A *enum* f) = f
proof
fix x
show *function-of* A *enum* (*sequence-of* A *enum* f) $x = f x$
using *assms* **unfolding** *function-of-def*
by (*auto simp add: length-sequence-of nth-sequence-of-inv-into*)
qed

4.3 Bijections

lemma *bij-betw-sequence-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\}$ A
shows *bij-betw* (*sequence-of* A *enum*) ($A \rightarrow_E B$) $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}$
proof (*rule bij-betw-byWitness*[**where** $f' = \text{function-of } A \text{ enum}$])
show $\forall f \in A \rightarrow_E B. \text{function-of } A \text{ enum } (\text{sequence-of } A \text{ enum } f) = f$
using *assms* **by** (*simp add: function-of-sequence-of*)
show $\forall xs \in \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}. \text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs) = xs$
using *assms* **by** (*auto simp add: sequence-of-function-of*)
show *sequence-of* A *enum* ' $(A \rightarrow_E B) \subseteq \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}$
using *assms* *set-sequence-of*[*OF assms*] *length-sequence-of* **by** *auto*
show *function-of* A *enum* ' $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\} \subseteq A \rightarrow_E B$
using *assms* *function-of-in-extensional-funcset* **by** *blast*
qed

lemma *bij-betw-function-of*:
assumes *bij-betw enum* $\{0..<\text{card } A\}$ A
shows *bij-betw* (*function-of* A *enum*) $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}$ ($A \rightarrow_E B$)
proof (*rule bij-betw-byWitness*[**where** $f' = \text{sequence-of } A \text{ enum}$])

show $\forall f \in A \rightarrow_E B$. *function-of A enum (sequence-of A enum f) = f*
using *assms by (simp add: function-of-sequence-of)*
show $\forall xs \in \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}$. *sequence-of A enum (function-of A enum xs) = xs*
using *assms by (auto simp add: sequence-of-function-of)*
show *sequence-of A enum ' (A \rightarrow_E B) \subseteq {xs. set xs \subseteq B \wedge length xs = card A}*
using *assms set-sequence-of[OF assms] length-sequence-of by auto*
show *function-of A enum ' {xs. set xs \subseteq B \wedge length xs = card A} \subseteq A \rightarrow_E B*
using *assms function-of-in-extensional-funcset by blast*
qed

4.4 Cardinality

lemma

assumes *finite A*

shows $\text{card } (A \rightarrow_E B) = \text{card } B \hat{\ } \text{card } A$

proof –

obtain *enum where bij-betw enum {0.. $\text{card } A$ } A*

using $\langle \text{finite } A \rangle$ *ex-bij-betw-nat-finite by blast*

have *bij-betw (sequence-of A enum) (A \rightarrow_E B) {xs. set xs \subseteq B \wedge length xs = card A}*

using $\langle \text{bij-betw enum } \{0..\text{card } A\} A \rangle$ *by (rule bij-betw-sequence-of)*

from this have $\text{card } (A \rightarrow_E B) = \text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\}$

by *(rule bij-betw-same-card)*

also have $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\} = \text{card } B \hat{\ } \text{card } A$

by *(rule card-lists-length-eq)*

finally show *?thesis .*

qed

lemma *card-sequences:*

assumes *finite A*

shows $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\} = \text{card } B \hat{\ } \text{card } A$

proof –

obtain *enum where bij-betw enum {0.. $\text{card } A$ } A*

using $\langle \text{finite } A \rangle$ *ex-bij-betw-nat-finite by blast*

have *bij-betw (function-of A enum) {xs. set xs \subseteq B \wedge length xs = card A} (A \rightarrow_E B)*

using $\langle \text{bij-betw enum } \{0..\text{card } A\} A \rangle$ *by (rule bij-betw-function-of)*

from this have $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A\} = \text{card } (A \rightarrow_E B)$

by *(rule bij-betw-same-card)*

also have $\text{card } (A \rightarrow_E B) = \text{card } B \hat{\ } \text{card } A$

using $\langle \text{finite } A \rangle$ *by (rule card-extensional-funcset)*

finally show *?thesis .*

qed

lemma

shows $\text{card } \{xs. \text{set } xs \subseteq A \wedge \text{length } xs = n\} = \text{card } A \hat{\ } n$

proof –

have $\text{card } \{xs. \text{set } xs \subseteq A \wedge \text{length } xs = n\} = \text{card } \{xs. \text{set } xs \subseteq A \wedge \text{length } xs$

```

= card {0..<n}
  by auto
  also have ... = card A ^ card {0..<n} by (subst card-sequences) auto
  also have ... = card A ^ n by auto
  finally show ?thesis .
qed

end

```

5 Injections from A to B

```

theory Twelfefold-Way-Entry2
imports Twelfefold-Way-Entry1
begin

```

Note that the cardinality theorems of both structures, distinct lists and finite injective functions, are already available. Hence, this development creates the bijection between those two structures and transfers the one cardinality theorem to the other structures and vice versa, although not strictly needed as both cardinality theorems were already available.

5.1 Properties for Bijections

```

lemma inj-on-implies-distinct:
  assumes bij-betw enum {0..<card A} A
  assumes f ∈ A →E B
  assumes inj-on f A
  shows distinct (sequence-of A enum f)
proof -
  {
    fix i j
    assume bounds: i < length (sequence-of A enum f) j < length (sequence-of A
enum f)
    assume i ≠ j
    from bounds assms(1, 2) have bounds': i < card A j < card A
    using length-sequence-of by fastforce+
    from this assms(1) have in-A: enum i ∈ A enum j ∈ A
    using bij-betwE by fastforce+
    from ⟨i ≠ j⟩ bounds' assms(1) have enum i ≠ enum j
    by (metis bij-betw-inv-into-left lessThan-iff atLeast0LessThan)
    from this have f (enum i) ≠ f (enum j)
    using assms(3) in-A inj-onD by fastforce
    from this bounds' have sequence-of A enum f ! i ≠ sequence-of A enum f ! j
    by (simp add: nth-sequence-of)
  }
  from this show ?thesis
  by (auto simp add: distinct-conv-nth)
qed

```

lemma *distinct-implies-inj-on*:
assumes *bij-betw enum* $\{0..<card\ A\}$ *A*
assumes *length xs = card A*
assumes *distinct xs*
shows *inj-on (function-of A enum xs) A*
proof (*rule inj-onI*)
let *?idx-of = $\lambda x. inv-into \{0..<length\ xs\}$ enum x*
fix *x y*
assume *x \in A y \in A function-of A enum xs x = function-of A enum xs y*
from *this* **have** *xs ! ?idx-of x = xs ! ?idx-of y*
unfolding *function-of-def* **by** *simp*
have *?idx-of x = ?idx-of y*
proof –
have *?idx-of x < length xs*
using $\langle x \in A \rangle$ *assms(1,2)*
by (*metis atLeast0LessThan bij-betw-imp-surj-on inv-into-into lessThan-iff*)
moreover **have** *?idx-of y < length xs*
using $\langle y \in A \rangle$ *assms(1,2)*
by (*metis atLeast0LessThan bij-betw-imp-surj-on inv-into-into lessThan-iff*)
moreover **note** $\langle xs ! ?idx-of x = xs ! ?idx-of y \rangle$ $\langle distinct\ xs \rangle$
ultimately **show** *?thesis*
by (*auto dest: nth-eq-iff-index-eq[where i=?idx-of x and j=?idx-of y]*)
qed
from *this* $\langle bij-betw\ -\ -\ \rightarrow \rangle$ **show** *x = y*
by (*metis $\langle x \in A \rangle \langle y \in A \rangle \langle length\ xs = card\ A \rangle bij-betw-inv-into-right$*)
qed

lemma *image-sequence-of-inj*:
assumes *bij-betw enum* $\{0..<card\ A\}$ *A*
shows *sequence-of A enum ‘ $\{f \in A \rightarrow_E B. inj-on\ f\ A\} \subseteq \{xs. set\ xs \subseteq B \wedge length\ xs = card\ A \wedge distinct\ xs\}$*
proof
fix *xs*
assume *xs \in sequence-of A enum ‘ $\{f \in A \rightarrow_E B. inj-on\ f\ A\}$*
from *this* **obtain** *f* **where** *xs: xs = sequence-of A enum f* **and** *f: f \in A \rightarrow_E B*
inj-on f A **by** *auto*
moreover **from** *xs f* $\langle bij-betw\ -\ -\ \rightarrow \rangle$ **have** *set xs \subseteq B*
using *set-sequence-of subsetCE* **by** *blast*
moreover **from** *xs f* $\langle bij-betw\ -\ -\ \rightarrow \rangle$ **have** *length xs = card A*
using *length-sequence-of* **by** *auto*
moreover **from** *xs f* $\langle bij-betw\ -\ -\ \rightarrow \rangle$ **have** *distinct xs*
using *inj-on-implies-distinct* **by** *simp*
ultimately **show** *xs \in $\{xs. set\ xs \subseteq B \wedge length\ xs = card\ A \wedge distinct\ xs\}$* **by**
auto
qed

lemma *image-function-of-distinct*:
assumes *bij-betw enum* $\{0..<card\ A\}$ *A*

shows *function-of A enum* ‘ $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
 $\subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A\}$
proof
fix f
assume $f: f \in \text{function-of } A \text{ enum } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
from $f \text{ assms}$ **have** $f \in A \rightarrow_E B$
using *function-of-in-extensional-funcset* **by** *blast*
moreover from $f \text{ assms}$ **have** *inj-on* $f A$
by (*auto simp add: assms distinct-implies-inj-on*)
ultimately show $f \in \{f \in A \rightarrow_E B. \text{inj-on } f A\}$ **by** *auto*
qed

5.2 Bijections

lemma *bij-betw-sequence-of*:

assumes *bij-betw enum* $\{0..<\text{card } A\} A$
shows *bij-betw (sequence-of A enum)* $\{f. f \in A \rightarrow_E B \wedge \text{inj-on } f A\} \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
proof (*rule bij-betw-byWitness* [**where** $f' = \text{function-of } A \text{ enum}$])
show $\forall f \in \{f \in A \rightarrow_E B. \text{inj-on } f A\}. \text{function-of } A \text{ enum } (\text{sequence-of } A \text{ enum } f) = f$
using *assms* **by** (*auto simp add: function-of-sequence-of*)
show $\forall xs \in \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}. \text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs) = xs$
using *assms* **by** (*auto simp add: sequence-of-function-of*)
show *sequence-of A enum* ‘ $\{f \in A \rightarrow_E B. \text{inj-on } f A\} \subseteq \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
using *assms* **by** (*simp add: image-sequence-of-inj*)
show *function-of A enum* ‘ $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\} \subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A\}$
using *assms* **by** (*simp add: image-function-of-distinct*)
qed

lemma *bij-betw-function-of*:

assumes *bij-betw enum* $\{0..<\text{card } A\} A$
shows *bij-betw (function-of A enum)* $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\} \{f \in A \rightarrow_E B. \text{inj-on } f A\}$
proof (*rule bij-betw-byWitness* [**where** $f' = \text{sequence-of } A \text{ enum}$])
show $\forall f \in \{f \in A \rightarrow_E B. \text{inj-on } f A\}. \text{function-of } A \text{ enum } (\text{sequence-of } A \text{ enum } f) = f$
using *assms* **by** (*auto simp add: function-of-sequence-of*)
show $\forall xs \in \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}. \text{sequence-of } A \text{ enum } (\text{function-of } A \text{ enum } xs) = xs$
using *assms* **by** (*auto simp add: sequence-of-function-of*)
show *sequence-of A enum* ‘ $\{f \in A \rightarrow_E B. \text{inj-on } f A\} \subseteq \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
using *assms* **by** (*simp add: image-sequence-of-inj*)
show *function-of A enum* ‘ $\{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$

$\subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A\}$
using *assms* **by** (*simp add: image-function-of-distinct*)
qed

5.3 Cardinality

lemma

assumes *finite A finite B card A ≤ card B*
shows $\text{card } \{f \in A \rightarrow_E B. \text{inj-on } f A\} = \prod \{\text{card } B - \text{card } A + 1 .. \text{card } B\}$
proof –
obtain *enum* **where** *bij-betw enum {0..<card A} A*
using $\langle \text{finite } A \rangle$ *ex-bij-betw-nat-finite* **by** *blast*
have *bij-betw (sequence-of A enum) {f ∈ A →_E B. inj-on f A} {xs. set xs ⊆ B*
 $\wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
using $\langle \text{bij-betw enum } \{0..<\text{card } A\} A \rangle$ **by** (*rule bij-betw-sequence-of*)
from *this* **have** $\text{card } \{f \in A \rightarrow_E B. \text{inj-on } f A\} = \text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\}$
by (*rule bij-betw-same-card*)
also **have** $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\} = \text{card } \{xs. \text{length } xs = \text{card } A \wedge \text{distinct } xs \wedge \text{set } xs \subseteq B\}$
by *meson*
also **have** $\text{card } \{xs. \text{length } xs = \text{card } A \wedge \text{distinct } xs \wedge \text{set } xs \subseteq B\} = \prod \{\text{card } B - \text{card } A + 1 .. \text{card } B\}$
using $\langle \text{finite } B \rangle$ $\langle \text{card } A \leq \text{card } B \rangle$ **by** (*rule List.card-lists-distinct-length-eq*)
finally **show** *?thesis* .
qed

lemma *card-sequences:*

assumes *finite A finite B card A ≤ card B*
shows $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\} = \text{fact } (\text{card } B) \text{ div fact } (\text{card } A)$
proof –
obtain *enum* **where** *bij-betw enum {0..<card A} A*
using $\langle \text{finite } A \rangle$ *ex-bij-betw-nat-finite* **by** *blast*
have *bij-betw (function-of A enum) {xs. set xs ⊆ B ∧ length xs = card A ∧ distinct xs} {f ∈ A →_E B. inj-on f A}*
using $\langle \text{bij-betw enum } \{0..<\text{card } A\} A \rangle$ **by** (*rule bij-betw-function-of*)
from *this* **have** $\text{card } \{xs. \text{set } xs \subseteq B \wedge \text{length } xs = \text{card } A \wedge \text{distinct } xs\} = \text{card } \{f \in A \rightarrow_E B. \text{inj-on } f A\}$
by (*rule bij-betw-same-card*)
also **have** $\text{card } \{f \in A \rightarrow_E B. \text{inj-on } f A\} = \text{fact } (\text{card } B) \text{ div fact } (\text{card } B - \text{card } A)$
using $\langle \text{finite } A \rangle$ $\langle \text{finite } B \rangle$ $\langle \text{card } A \leq \text{card } B \rangle$ **by** (*rule card-extensional-funcset-inj-on*)
finally **show** *?thesis* .
qed

end

6 Functions from A to B, up to a Permutation of A

theory *Twelvefold-Way-Entry4*
imports *Equiv-Relations-on-Functions*
begin

6.1 Definition of Bijections

definition *msubset-of* :: 'a set \Rightarrow ('a \Rightarrow 'b) set \Rightarrow 'b multiset
where

msubset-of A F = univ (λ f. image-mset f (mset-set A)) F

definition *functions-of* :: 'a set \Rightarrow 'b multiset \Rightarrow ('a \Rightarrow 'b) set
where

functions-of A B = {f \in A \rightarrow_E set-mset B. image-mset f (mset-set A) = B}

6.2 Properties for Bijections

lemma *msubset-of*:

assumes F \in (A \rightarrow_E B) // domain-permutation A B

shows size (msubset-of A F) = card A

and set-mset (msubset-of A F) \subseteq B

proof –

from \langle F \in (A \rightarrow_E B) // domain-permutation A B \rangle **obtain** f **where** f \in A \rightarrow_E B

and F-eq: F = domain-permutation A B “ {f} **using** quotientE **by** blast

have msubset-of A F = univ (λ f. image-mset f (mset-set A)) F

unfolding msubset-of-def ..

also have ... = univ (λ f. image-mset f (mset-set A)) (domain-permutation A B “ {f})

unfolding F-eq ..

also have ... = image-mset f (mset-set A)

using equiv-domain-permutation image-mset-respects-domain-permutation \langle f \in A \rightarrow_E B \rangle

by (subst univ-commute') auto

finally have msubset-of-eq: msubset-of A F = image-mset f (mset-set A) .

show size (msubset-of A F) = card A

proof –

have size (msubset-of A F) = size (image-mset f (mset-set A))

unfolding msubset-of-eq ..

also have ... = card A

by (cases \langle finite A \rangle) auto

finally show ?thesis .

qed

show set-mset (msubset-of A F) \subseteq B

proof –

have set-mset (msubset-of A F) = set-mset (image-mset f (mset-set A))

unfolding msubset-of-eq ..

```

    also have ...  $\subseteq$  B
      using  $\langle f \in A \rightarrow_E B \rangle$  by (cases finite A) auto
    finally show ?thesis .
  qed
qed

lemma functions-of:
  assumes finite A
  assumes set-mset M  $\subseteq$  B
  assumes size M = card A
  shows functions-of A M  $\in$  (A  $\rightarrow_E$  B) // domain-permutation A B
proof -
  obtain f where f  $\in$  A  $\rightarrow_E$  set-mset M and image-mset f (mset-set A) = M
    using obtain-function-on-ext-funcset  $\langle$ finite A $\rangle$   $\langle$ size M = card A $\rangle$  by blast
  from  $\langle f \in A \rightarrow_E$  set-mset M $\rangle$  have f  $\in$  A  $\rightarrow_E$  B
    using  $\langle$ set-mset M  $\subseteq$  B $\rangle$  PiE-iff subset-eq by blast
  have functions-of A M = (domain-permutation A B) “ {f}
  proof
    show functions-of A M  $\subseteq$  domain-permutation A B “ {f}
    proof
      fix f'
      assume f'  $\in$  functions-of A M
      from this have M = image-mset f' (mset-set A) and f'  $\in$  A  $\rightarrow_E$  f' ‘ A
        using  $\langle$ finite A $\rangle$  unfolding functions-of-def by auto
      from this assms(1, 2) have f'  $\in$  A  $\rightarrow_E$  B
        by (simp add: PiE-iff image-subset-iff)
      obtain p where p permutes A  $\wedge$  ( $\forall x \in A. f x = f' (p x)$ )
        using  $\langle$ finite A $\rangle$   $\langle$ image-mset f (mset-set A) = M $\rangle$   $\langle$ M = image-mset f'
        (mset-set A) $\rangle$ 
        image-mset-eq-implies-permutes by blast
      from this show f'  $\in$  domain-permutation A B “ {f}
        using  $\langle f \in A \rightarrow_E B \rangle$   $\langle f' \in A \rightarrow_E B \rangle$ 
        unfolding domain-permutation-def by auto
    qed
  next
    show domain-permutation A B “ {f}  $\subseteq$  functions-of A M
    proof
      fix f'
      assume f'  $\in$  domain-permutation A B “ {f}
      from this have (f, f')  $\in$  domain-permutation A B by auto
      from this  $\langle$ image-mset f (mset-set A) = M $\rangle$  have image-mset f' (mset-set A)
      = M
        using congruentD[OF image-mset-respects-domain-permutation] by metis
      moreover from this  $\langle$ (f, f')  $\in$  domain-permutation A B $\rangle$  have f'  $\in$  A  $\rightarrow_E$ 
      set-mset M
        using  $\langle$ finite A $\rangle$  unfolding domain-permutation-def by auto
      ultimately show f'  $\in$  functions-of A M
        unfolding functions-of-def by auto
    qed
  qed

```

qed
from $\langle f \in A \rightarrow_E B \rangle$ **show** *?thesis* **by** (*auto intro: quotientI*)
qed

lemma *functions-of-msubset-of*:
assumes *finite A*
assumes $F \in (A \rightarrow_E B)$ // *domain-permutation A B*
shows *functions-of A (msubset-of A F) = F*
proof –
from $\langle F \in (A \rightarrow_E B)$ // *domain-permutation A B* **obtain** f **where** $f \in A \rightarrow_E B$
and F -*eq*: $F = \text{domain-permutation } A \ B \ \{f\}$ **using** *quotientE* **by** *blast*
have $\text{msubset-of } A \ F = \text{univ } (\lambda f. \text{image-mset } f \ (\text{mset-set } A)) \ F$
unfolding *msubset-of-def* ..
also have $\dots = \text{univ } (\lambda f. \text{image-mset } f \ (\text{mset-set } A)) \ (\text{domain-permutation } A \ B \ \{f\})$
unfolding F -*eq* ..
also have $\dots = \text{image-mset } f \ (\text{mset-set } A)$
using *equiv-domain-permutation image-mset-respects-domain-permutation* $\langle f \in A \rightarrow_E B \rangle$
by (*subst univ-commute'*) *auto*
finally have $\text{msubset-of-eq: } \text{msubset-of } A \ F = \text{image-mset } f \ (\text{mset-set } A)$.
show *?thesis*
proof
show *functions-of A (msubset-of A F) \subseteq F*
proof
fix f'
assume $f' \in \text{functions-of } A \ (\text{msubset-of } A \ F)$
from *this* **have** $f': f' \in A \rightarrow_E f' \ \text{set-mset } (\text{mset-set } A)$
 $\text{image-mset } f' \ (\text{mset-set } A) = \text{image-mset } f \ (\text{mset-set } A)$
unfolding *functions-of-def* **by** (*auto simp add: msubset-of-eq*)
from $\langle f \in A \rightarrow_E B \rangle$ **have** $f' \ A \subseteq B$ **by** *auto*
note $\langle f \in A \rightarrow_E B \rangle$
moreover from $f'(1) \ \langle \text{finite } A \rangle \ \langle f' \ A \subseteq B \rangle$ **have** $f' \in A \rightarrow_E B$ **by** *auto*
moreover obtain p **where** p *permutes* $A \wedge (\forall x \in A. f \ x = f' (p \ x))$
using $\langle \text{finite } A \rangle \ \langle \text{image-mset } f' \ (\text{mset-set } A) = \text{image-mset } f \ (\text{mset-set } A) \rangle$
by (*metis image-mset-eq-implies-permutes*)
ultimately show $f' \in F$
unfolding F -*eq domain-permutation-def* **by** *auto*
qed

next
show $F \subseteq \text{functions-of } A \ (\text{msubset-of } A \ F)$
proof
fix f'
assume $f' \in F$
from *this* **have** $f' \in A \rightarrow_E B$
unfolding F -*eq domain-permutation-def* **by** *auto*
from $\langle f' \in F \rangle$ **obtain** p **where** p *permutes* $A \wedge (\forall x \in A. f \ x = f' (p \ x))$
unfolding F -*eq domain-permutation-def* **by** *auto*

from this have $eq: image\text{-}mset\ f' (mset\text{-}set\ A) = image\text{-}mset\ f (mset\text{-}set\ A)$
using *permutates-implies-image-mset-eq* **by** *blast*
moreover have $f' \in A \rightarrow_E\ set\text{-}mset\ (image\text{-}mset\ f (mset\text{-}set\ A))$
using $\langle finite\ A \rangle \langle f' \in A \rightarrow_E\ B \rangle eq[symmetric]$ **by** *auto*
ultimately show $f' \in functions\text{-}of\ A (msubset\text{-}of\ A\ F)$
unfolding *functions-of-def msubset-of-eq* **by** *auto*
qed
qed
qed

lemma *msubset-of-functions-of*:

assumes $set\text{-}mset\ M \subseteq B\ size\ M = card\ A\ finite\ A$
shows $msubset\text{-}of\ A (functions\text{-}of\ A\ M) = M$
proof $-$
from *assms* **have** $functions\text{-}of\ A\ M \in (A \rightarrow_E\ B) // domain\text{-}permutation\ A\ B$
using *functions-of* **by** *fastforce*
from this obtain f **where** $f \in A \rightarrow_E\ B$ **and** $functions\text{-}of\ A\ M = domain\text{-}permutation\ A\ B\ \{\{f\}\}$
by *(rule quotientE)*
from this have $f \in functions\text{-}of\ A\ M$
using *equiv-domain-permutation equiv-class-self* **by** *fastforce*
have $msubset\text{-}of\ A (functions\text{-}of\ A\ M) = univ\ (\lambda f. image\text{-}mset\ f (mset\text{-}set\ A))$
(functions-of A M)
unfolding *msubset-of-def ..*
also have $\dots = univ\ (\lambda f. image\text{-}mset\ f (mset\text{-}set\ A)) (domain\text{-}permutation\ A\ B\ \{\{f\}\})$
unfolding $\langle functions\text{-}of\ A\ M = domain\text{-}permutation\ A\ B\ \{\{f\}\} \dots$
also have $\dots = image\text{-}mset\ f (mset\text{-}set\ A)$
using *equiv-domain-permutation image-mset-respects-domain-permutation* $\langle f \in A \rightarrow_E\ B \rangle$
by *(subst univ-commute')* *auto*
also have $image\text{-}mset\ f (mset\text{-}set\ A) = M$
using $\langle f \in functions\text{-}of\ A\ M \rangle$ **unfolding** *functions-of-def* **by** *simp*
finally show *?thesis .*
qed

6.3 Bijections

lemma *bij-betw-msubset-of*:

assumes *finite A*
shows $bij\text{-}betw (msubset\text{-}of\ A) ((A \rightarrow_E\ B) // domain\text{-}permutation\ A\ B) \{M. set\text{-}mset\ M \subseteq B \wedge size\ M = card\ A\}$
proof *(rule bij-betw-byWitness[where f'=λM. functions-of A M])*
show $\forall F \in (A \rightarrow_E\ B) // domain\text{-}permutation\ A\ B. functions\text{-}of\ A (msubset\text{-}of\ A\ F) = F$
using $\langle finite\ A \rangle$ **by** *(auto simp add: functions-of-msubset-of)*
show $\forall M \in \{M. set\text{-}mset\ M \subseteq B \wedge size\ M = card\ A\}. msubset\text{-}of\ A (functions\text{-}of\ A\ M) = M$
using $\langle finite\ A \rangle$ **by** *(auto simp add: msubset-of-functions-of)*

```

show msubset-of A ‘ ((A →E B) // domain-permutation A B) ⊆ {M. set-mset
M ⊆ B ∧ size M = card A}
  using msubset-of by blast
show functions-of A ‘ {M. set-mset M ⊆ B ∧ size M = card A} ⊆ (A →E B)
// domain-permutation A B
  using functions-of ⟨finite A⟩ by blast
qed

```

6.4 Cardinality

lemma

```

assumes finite A finite B
shows card ((A →E B) // domain-permutation A B) = card B + card A - 1
choose card A
proof -
  have bij-betw (msubset-of A) ((A →E B) // domain-permutation A B) {M.
set-mset M ⊆ B ∧ size M = card A}
    using ⟨finite A⟩ by (rule bij-betw-msubset-of)
  from this have card ((A →E B) // domain-permutation A B) = card {M.
set-mset M ⊆ B ∧ size M = card A}
    by (rule bij-betw-same-card)
  also have card {M. set-mset M ⊆ B ∧ size M = card A} = card B + card A -
1 choose card A
    using ⟨finite B⟩ by (rule card-multisets)
  finally show ?thesis .
qed

```

end

7 Injections from A to B up to a Permutation of A

theory *Twelvefold-Way-Entry5*

imports

Equiv-Relations-on-Functions

begin

7.1 Definition of Bijections

definition *subset-of* :: 'a set ⇒ ('a ⇒ 'b) set ⇒ 'b set

where

subset-of A F = *univ* (λf. f ‘ A) F

definition *functions-of* :: 'a set ⇒ 'b set ⇒ ('a ⇒ 'b) set

where

functions-of A B = {f ∈ A →_E B. f ‘ A = B}

7.2 Properties for Bijections

lemma *functions-of-eq*:

assumes *finite A*

assumes $f \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$

shows *functions-of A (f ' A) = domain-permutation A B “ {f}*

proof

have *bij: bij-betw f A (f ' A)*

using *assms by (simp add: bij-betw-imageI)*

show *functions-of A (f ' A) \subseteq domain-permutation A B “ {f}*

proof

fix *f'*

assume $f' \in \text{functions-of } A \ (f' \ A)$

from *this* have $f' \in A \rightarrow_E f' \ A$ and $f' \ ' \ A = f' \ A$

unfolding *functions-of-def by auto*

from *this* *assms* have $f' \in A \rightarrow_E B$ and *inj-on f A*

using *PiE-mem by fastforce+*

moreover have $\exists p. p \text{ permutes } A \wedge (\forall x \in A. f \ x = f' \ (p \ x))$

proof

let $?p = \lambda x. \text{if } x \in A \text{ then } \text{inv-into } A \ f' \ (f \ x) \text{ else } x$

show $?p \text{ permutes } A \wedge (\forall x \in A. f \ x = f' \ (?p \ x))$

proof

show $?p \text{ permutes } A$

proof (*rule bij-imp-permutes*)

show *bij-betw ?p A A*

proof (*rule bij-betw-imageI*)

show *inj-on ?p A*

proof (*rule inj-onI*)

fix *a a'*

assume $a \in A \ a' \in A \ ?p \ a = ?p \ a'$

from *this* have *inv-into A f' (f a) = inv-into A f' (f a')* **by** *auto*

from *this* $\langle a \in A \rangle \langle a' \in A \rangle \langle f' \ ' \ A = f' \ A \rangle$ have $f \ a = f \ a'$

using *inv-into-injective by fastforce*

from *this* $\langle a \in A \rangle \langle a' \in A \rangle$ show $a = a'$

by (*metis bij bij-betw-inv-into-left*)

qed

next

show $?p \ ' \ A = A$

proof

show $?p \ ' \ A \subseteq A$

using $\langle f' \ ' \ A = f' \ A \rangle$ **by** (*simp add: image-subsetI inv-into-into*)

next

show $A \subseteq ?p \ ' \ A$

proof

fix *a*

assume $a \in A$

have *inj-on f' A*

using $\langle \text{finite } A \rangle \langle f' \ ' \ A = f' \ A \rangle \langle \text{inj-on } f \ A \rangle$

by (*simp add: card-image eq-card-imp-inj-on*)

from $\langle a \in A \rangle \langle f' \ ' \ A = f' \ A \rangle$ have *inv-into A f (f' a) \in A*

```

      by (metis image-eqI inv-into-into)
    moreover have  $a = \text{inv-into } A \ f' \ (f \ (\text{inv-into } A \ f \ (f' \ a)))$ 
      using  $\langle a \in A \rangle \langle f' \ ' \ A = f \ ' \ A \rangle \langle \text{inj-on } f' \ A \rangle$ 
      by (metis f-inv-into-f image-eqI inv-into-f-f)
    ultimately show  $a \in ?p \ ' \ A$  by auto
  qed
qed
next
fix  $x$ 
assume  $x \notin A$ 
from this show  $?p \ x = x$  by simp
qed
next
from  $\langle f' \ ' \ A = f \ ' \ A \rangle$  show  $\forall x \in A. f \ x = f' \ (?p \ x)$ 
  by (simp add: f-inv-into-f)
qed
moreover have  $f \in A \rightarrow_E \ B$  using assms by auto
ultimately show  $f' \in \text{domain-permutation } A \ B \ \{\{f\}\}$ 
  unfolding domain-permutation-def by auto
qed
next
show  $\text{domain-permutation } A \ B \ \{\{f\}\} \subseteq \text{functions-of } A \ (f \ ' \ A)$ 
proof
fix  $f'$ 
assume  $f' \in \text{domain-permutation } A \ B \ \{\{f\}\}$ 
from this obtain  $p$  where  $p$  permutes  $A \ \forall x \in A. f \ x = f' \ (p \ x)$ 
  and  $f \in A \rightarrow_E \ B \ f' \in A \rightarrow_E \ B$ 
  unfolding domain-permutation-def by auto
have  $f' \ ' \ A = f \ ' \ A$ 
proof
show  $f' \ ' \ A \subseteq f \ ' \ A$ 
proof
fix  $x$ 
assume  $x \in f' \ ' \ A$ 
from this obtain  $x'$  where  $x = f' \ x'$  and  $x' \in A$  ..
from this have  $x = f \ (\text{inv } p \ x')$ 
using  $p$  by (metis (mono-tags, lifting) permutes-in-image permutes-inverses(1))
moreover have  $\text{inv } p \ x' \in A$ 
  using  $p \ \langle x' \in A \rangle$  by (simp add: permutes-in-image permutes-inv)
ultimately show  $x \in f \ ' \ A$  ..
qed
next
show  $f \ ' \ A \subseteq f' \ ' \ A$ 
  using  $p$  permutes-in-image by fastforce
qed
moreover from this  $\langle f' \in A \rightarrow_E \ B \rangle$  have  $f' \in A \rightarrow_E \ f' \ ' \ A$  by auto
ultimately show  $f' \in \text{functions-of } A \ (f \ ' \ A)$ 

```

unfolding functions-of-def by auto
qed
qed

lemma subset-of:

assumes $F \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$ // domain-permutation $A \ B$
shows $\text{subset-of } A \ F \subseteq B$ **and** $\text{card } (\text{subset-of } A \ F) = \text{card } A$

proof –

from *assms* **obtain** f **where** $F\text{-eq: } F = (\text{domain-permutation } A \ B) \ \{\!\{f\}\!\}$
and $f: f \in A \rightarrow_E B \ \text{inj-on } f \ A$
using *mem-Collect-eq quotientE* **by** *force*
from *this* **have** $\text{subset-of } A \ (\text{domain-permutation } A \ B \ \{\!\{f\}\!\}) = f \ \text{' } A$
using *equiv-domain-permutation image-respects-domain-permutation*
unfolding *subset-of-def* **by** (*intro univ-commute'*) *auto*
from *this* $f \ F\text{-eq}$ **show** $\text{subset-of } A \ F \subseteq B$ **and** $\text{card } (\text{subset-of } A \ F) = \text{card } A$
by (*auto simp add: card-image*)

qed

lemma functions-of:

assumes *finite* A *finite* B $X \subseteq B$ $\text{card } X = \text{card } A$
shows $\text{functions-of } A \ X \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$ // domain-permutation $A \ B$

proof –

from *assms* **obtain** f **where** $f: f \in A \rightarrow_E X \wedge \text{bij-betw } f \ A \ X$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*metis finite-same-card-bij-on-ext-funcset finite-subset*)
from *this* **have** $X = f \ \text{' } A$ **by** (*simp add: bij-betw-def*)
from $f \ \langle X \subseteq B \rangle$ **have** $f \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$
by (*auto simp add: bij-betw-imp-inj-on*)
have $\text{functions-of } A \ X = \text{domain-permutation } A \ B \ \{\!\{f\}\!\}$
using $\langle \text{finite } A \rangle \langle X = f \ \text{' } A \rangle \langle f \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\} \rangle$
by (*simp add: functions-of-eq*)
from *this* **show** $\text{functions-of } A \ X \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$ // domain-permutation $A \ B$
using $\langle f \in \{f \in A \rightarrow_E B. \text{inj-on } f \ A\} \rangle$ **by** (*auto intro: quotientI*)

qed

lemma subset-of-functions-of:

assumes *finite* A *finite* X $\text{card } A = \text{card } X$
shows $\text{subset-of } A \ (\text{functions-of } A \ X) = X$

proof –

from *assms* **obtain** f **where** $f \in A \rightarrow_E X$ **and** $\text{bij-betw } f \ A \ X$
using *finite-same-card-bij-on-ext-funcset* **by** *blast*
from *this* **have** $\text{subset-of: } \text{subset-of } A \ (\text{domain-permutation } A \ X \ \{\!\{f\}\!\}) = f \ \text{' } A$
using *equiv-domain-permutation image-respects-domain-permutation*
unfolding *subset-of-def* **by** (*intro univ-commute'*) *auto*
from $\langle \text{bij-betw } f \ A \ X \rangle$ **have** $\text{inj-on } f \ A$ **and** $f \ \text{' } A = X$
by (*auto simp add: bij-betw-def*)
have $\text{subset-of } A \ (\text{functions-of } A \ X) = \text{subset-of } A \ (\text{functions-of } A \ (f \ \text{' } A))$


```

    using ⟨f ‘ A = X⟩ by simp
  also have ... = subset-of A (domain-permutation A X “ {f})
    using ⟨finite A⟩ ⟨inj-on f A⟩ ⟨f ∈ A →E X⟩ by (auto simp add: functions-of-eq)
  also have ... = f ‘ A
    using ⟨inj-on f A⟩ ⟨f ∈ A →E X⟩ by (simp add: subset-of)
  also have ... = X
    using ⟨f ‘ A = X⟩ by simp
  finally show ?thesis .
qed

```

lemma *functions-of-subset-of*:

```

  assumes finite A
  assumes F ∈ {f ∈ A →E B. inj-on f A} // domain-permutation A B
  shows functions-of A (subset-of A F) = F
using assms(2) proof (rule quotientE)
  fix f
  assume f: f ∈ {f ∈ A →E B. inj-on f A}
    and F-eq: F = domain-permutation A B “ {f}
  from this have subset-of A (domain-permutation A B “ {f}) = f ‘ A
    using equiv-domain-permutation image-respects-domain-permutation
    unfolding subset-of-def by (intro univ-commute^) auto
  from this f F-eq ⟨finite A⟩ show functions-of A (subset-of A F) = F
    by (simp add: functions-of-eq)
qed

```

7.3 Bijections

lemma *bij-betw-subset-of*:

```

  assumes finite A finite B
  shows bij-betw (subset-of A) ({f ∈ A →E B. inj-on f A} // domain-permutation
  A B) {X. X ⊆ B ∧ card X = card A}
proof (rule bij-betw-byWitness[where f'=functions-of A])
  show ∀ F ∈ {f ∈ A →E B. inj-on f A} // domain-permutation A B. functions-of
  A (subset-of A F) = F
    using ⟨finite A⟩ functions-of-subset-of by auto
  show ∀ X ∈ {X. X ⊆ B ∧ card X = card A}. subset-of A (functions-of A X) = X
    using subset-of-functions-of ⟨finite A⟩ ⟨finite B⟩
    by (metis (mono-tags) finite-subset mem-Collect-eq)
  show subset-of A ‘ ({f ∈ A →E B. inj-on f A} // domain-permutation A B) ⊆
  {X. X ⊆ B ∧ card X = card A}
    using subset-of by fastforce
  show functions-of A ‘ {X. X ⊆ B ∧ card X = card A} ⊆ {f ∈ A →E B. inj-on
  f A} // domain-permutation A B
    using ⟨finite A⟩ ⟨finite B⟩ functions-of by auto
qed

```

lemma *bij-betw-functions-of*:

```

  assumes finite A finite B
  shows bij-betw (functions-of A) {X. X ⊆ B ∧ card X = card A} ({f ∈ A →E

```

$B. \text{inj-on } f A \} // \text{domain-permutation } A B)$
proof (rule *bij-betw-byWitness*[**where** $f' = \text{subset-of } A$])
show $\forall F \in \{f \in A \rightarrow_E B. \text{inj-on } f A \} // \text{domain-permutation } A B. \text{functions-of } A (\text{subset-of } A F) = F$
using $\langle \text{finite } A \rangle \text{functions-of-subset-of}$ **by** *auto*
show $\forall X \in \{X. X \subseteq B \wedge \text{card } X = \text{card } A \}. \text{subset-of } A (\text{functions-of } A X) = X$
using *subset-of-functions-of* $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$
by (*metis (mono-tags) finite-subset mem-Collect-eq*)
show $\text{subset-of } A ' (\{f \in A \rightarrow_E B. \text{inj-on } f A \} // \text{domain-permutation } A B) \subseteq \{X. X \subseteq B \wedge \text{card } X = \text{card } A \}$
using *subset-of* **by** *fastforce*
show $\text{functions-of } A ' \{X. X \subseteq B \wedge \text{card } X = \text{card } A \} \subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A \} // \text{domain-permutation } A B$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle \text{functions-of}$ **by** *auto*
qed

lemma *bij-betw-mset-set*:

shows $\text{bij-betw mset-set } \{A. \text{finite } A \} \{M. \forall x. \text{count } M x \leq 1 \}$
proof (rule *bij-betw-byWitness*[**where** $f' = \text{set-mset}$])
show $\forall A \in \{A. \text{finite } A \}. \text{set-mset } (\text{mset-set } A) = A$ **by** *auto*
show $\forall M \in \{M. \forall x. \text{count } M x \leq 1 \}. \text{mset-set } (\text{set-mset } M) = M$
by (*auto simp add: mset-set-set-mset'*)
show $\text{mset-set } ' \{A. \text{finite } A \} \subseteq \{M. \forall x. \text{count } M x \leq 1 \}$
using *nat-le-linear* **by** *fastforce*
show $\text{set-mset } ' \{M. \forall x. \text{count } M x \leq 1 \} \subseteq \{A. \text{finite } A \}$ **by** *auto*
qed

lemma *bij-betw-mset-set-card*:

assumes *finite A*
shows $\text{bij-betw mset-set } \{X. X \subseteq A \wedge \text{card } X = k \} \{M. M \subseteq \# \text{mset-set } A \wedge \text{size } M = k \}$
proof (rule *bij-betw-byWitness*[**where** $f' = \text{set-mset}$])
show $\forall X \in \{X. X \subseteq A \wedge \text{card } X = k \}. \text{set-mset } (\text{mset-set } X) = X$
using $\langle \text{finite } A \rangle \text{rev-finite-subset[of } A \rangle$ **by** *auto*
show $\forall M \in \{M. M \subseteq \# \text{mset-set } A \wedge \text{size } M = k \}. \text{mset-set } (\text{set-mset } M) = M$
by (*auto simp add: mset-set-set-mset'*)
show $\text{mset-set } ' \{X. X \subseteq A \wedge \text{card } X = k \} \subseteq \{M. M \subseteq \# \text{mset-set } A \wedge \text{size } M = k \}$
using $\langle \text{finite } A \rangle \text{rev-finite-subset[of } A \rangle$
by (*auto simp add: mset-set-subseteq-mset-set*)
show $\text{set-mset } ' \{M. M \subseteq \# \text{mset-set } A \wedge \text{size } M = k \} \subseteq \{X. X \subseteq A \wedge \text{card } X = k \}$
using *assms mset-subset-eqD card-set-mset* **by** *fastforce*
qed

lemma *bij-betw-mset-set-card'*:

assumes *finite A*
shows $\text{bij-betw mset-set } \{X. X \subseteq A \wedge \text{card } X = k \} \{M. \text{set-mset } M \subseteq A \wedge \text{size } M = k \wedge (\forall x. \text{count } M x \leq 1) \}$

proof (rule *bij-betw-byWitness*[**where** $f' = \text{set-mset}$])
show $\forall X \in \{X. X \subseteq A \wedge \text{card } X = k\}. \text{set-mset } (\text{mset-set } X) = X$
using $\langle \text{finite } A \rangle \text{ rev-finite-subset[of } A]$ **by** *auto*
show $\forall M \in \{M. \text{set-mset } M \subseteq A \wedge \text{size } M = k \wedge (\forall x. \text{count } M \ x \leq 1)\}. \text{mset-set } (\text{set-mset } M) = M$
by (*auto simp add: mset-set-set-mset'*)
show $\text{mset-set } \{X. X \subseteq A \wedge \text{card } X = k\} \subseteq \{M. \text{set-mset } M \subseteq A \wedge \text{size } M = k \wedge (\forall x. \text{count } M \ x \leq 1)\}$
using $\langle \text{finite } A \rangle \text{ rev-finite-subset[of } A]$ **by** (*auto simp add: count-mset-set-leq'*)
show $\text{set-mset } \{M. \text{set-mset } M \subseteq A \wedge \text{size } M = k \wedge (\forall x. \text{count } M \ x \leq 1)\} \subseteq \{X. X \subseteq A \wedge \text{card } X = k\}$
by (*auto simp add: card-set-mset'*)
qed

7.4 Cardinality

lemma *card-injective-functions-domain-permutation*:

assumes *finite A finite B*

shows $\text{card } (\{f \in A \rightarrow_E B. \text{inj-on } f \ A\} // \text{domain-permutation } A \ B) = \text{card } B$
choose card A

proof –

have *bij-betw (subset-of A) (f ∈ A →_E B. inj-on f A) // domain-permutation A B* $\{X. X \subseteq B \wedge \text{card } X = \text{card } A\}$

using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*rule bij-betw-subset-of*)

from this have $\text{card } (\{f \in A \rightarrow_E B. \text{inj-on } f \ A\} // \text{domain-permutation } A \ B) = \text{card } \{X. X \subseteq B \wedge \text{card } X = \text{card } A\}$

by (*rule bij-betw-same-card*)

also have $\text{card } \{X. X \subseteq B \wedge \text{card } X = \text{card } A\} = \text{card } B$ *choose card A*

using $\langle \text{finite } B \rangle$ **by** (*rule n-subsets*)

finally show *?thesis .*

qed

lemma *card-multiset-only-sets*:

assumes *finite A*

shows $\text{card } \{M. M \subseteq\# \text{mset-set } A \wedge \text{size } M = k\} = \text{card } A$ *choose k*

proof –

have *bij-betw mset-set {X. X ⊆ A ∧ card X = k} {M. M ⊆# mset-set A ∧ size M = k}*

using $\langle \text{finite } A \rangle$ **by** (*rule bij-betw-mset-set-card*)

from this have $\text{card } \{M. M \subseteq\# \text{mset-set } A \wedge \text{size } M = k\} = \text{card } \{X. X \subseteq A \wedge \text{card } X = k\}$

by (*simp add: bij-betw-same-card*)

also have $\text{card } \{X. X \subseteq A \wedge \text{card } X = k\} = \text{card } A$ *choose k*

using $\langle \text{finite } A \rangle$ **by** (*rule n-subsets*)

finally show *?thesis .*

qed

lemma *card-multiset-only-sets'*:

assumes *finite A*

```

shows  $\text{card } \{M. \text{ set-mset } M \subseteq A \wedge \text{ size } M = k \wedge (\forall x. \text{ count } M x \leq 1)\} = \text{card } A$ 
choose  $k$ 
proof –
  from  $\langle \text{finite } A \rangle$  have  $\{M. \text{ set-mset } M \subseteq A \wedge \text{ size } M = k \wedge (\forall x. \text{ count } M x \leq 1)\} =$ 
     $\{M. M \subseteq \# \text{ mset-set } A \wedge \text{ size } M = k\}$ 
  using msubset-mset-set-iff by auto
  from this  $\langle \text{finite } A \rangle$  card-multiset-only-sets show ?thesis by simp
qed

end

```

8 Surjections from A to B up to a Permutation on A

```

theory Twelvefold-Way-Entry6
imports Twelvefold-Way-Entry4
begin

```

8.1 Properties for Bijections

```

lemma set-mset-eq-implies-surj-on:
  assumes finite A
  assumes  $\text{size } M = \text{card } A$  set-mset  $M = B$ 
  assumes  $f \in \text{functions-of } A M$ 
  shows  $f ' A = B$ 
proof –
  from  $\langle f \in \text{functions-of } A M \rangle$  have image-mset  $f$  (mset-set  $A$ ) =  $M$ 
  unfolding functions-of-def by auto
  from  $\langle \text{image-mset } f$  (mset-set  $A$ ) =  $M \rangle$  show  $f ' A = B$ 
  using  $\langle \text{set-mset } M = B \rangle$   $\langle \text{finite } A \rangle$  finite-set-mset-mset-set set-image-mset by
force
qed

```

```

lemma surj-on-implies-set-mset-eq:
  assumes finite A
  assumes  $F \in (A \rightarrow_E B)$  // domain-permutation A B
  assumes univ  $(\lambda f. f ' A = B)$   $F$ 
  shows set-mset (msubset-of  $A F$ ) =  $B$ 
proof –
  from  $\langle F \in (A \rightarrow_E B)$  // domain-permutation A B  $\rangle$  obtain  $f$  where  $f \in A \rightarrow_E B$ 
  and F-eq:  $F = \text{domain-permutation } A B$  “ $\{f\}$ ” using quotientE by blast
  have msubset-of  $A F = \text{univ } (\lambda f. \text{image-mset } f$  (mset-set  $A$ ))  $F$ 
  unfolding msubset-of-def ..
  also have  $\dots = \text{univ } (\lambda f. \text{image-mset } f$  (mset-set  $A$ )) (domain-permutation A B
  “ $\{f\}$ ”)
  unfolding F-eq ..

```

```

also have ... = image-mset f (mset-set A)
using equiv-domain-permutation image-mset-respects-domain-permutation ⟨f ∈
A →E B⟩
by (subst univ-commute') auto
finally have eq: msubset-of A F = image-mset f (mset-set A) .
from iffD1[OF univ-commute', OF equiv-domain-permutation, OF surjective-respects-domain-permutation,
OF ⟨f ∈ A →E B⟩
```

⟨*univ* (λ*f*. *f* ‘ *A* = *B*) *F*⟩ **have** *f* ‘ *A* = *B* **by** (*simp add: F-eq*)
have *set-mset (image-mset f (mset-set A)) = B*
proof
show *set-mset (image-mset f (mset-set A)) ⊆ B*
using ⟨*finite A*⟩ *f* ‘ *A* = *B* **by** *auto*
next
show *B ⊆ set-mset (image-mset f (mset-set A))*
using ⟨*finite A*⟩ **by** (*simp add: f* ‘ *A* = *B*)[*symmetric*] *in-image-mset*)
qed
from *this* **show** *set-mset (msubset-of A F) = B*
unfolding *eq* .
qed

lemma *functions-of-is-surj-on:*

```

assumes finite A
assumes size M = card A set-mset M = B
shows univ (λf. f ‘ A = B) (functions-of A M)
proof –
have functions-of A M ∈ (A →E B) // domain-permutation A B
using functions-of ⟨finite A⟩ ⟨size M = card A⟩ ⟨set-mset M = B⟩ by fastforce
from this obtain f where eq-f: functions-of A M = domain-permutation A B
“{f}” and f ∈ A →E B
using quotientE by blast
from eq-f have f ∈ functions-of A M
using ⟨f ∈ A →E B⟩ equiv-domain-permutation equiv-class-self by fastforce
have f ‘ A = B
using ⟨f ∈ functions-of A M⟩ assms set-mset-eq-implies-surj-on by fastforce
from this show ?thesis
unfolding eq-f using equiv-domain-permutation surjective-respects-domain-permutation
⟨f ∈ A →E B⟩
by (subst univ-commute') assumption+
qed

```

8.2 Bijections

lemma *bij-betw-msubset-of:*

```

assumes finite A
shows bij-betw (msubset-of A) ({f ∈ A →E B. f ‘ A = B} // domain-permutation
A B)
  {M. set-mset M = B ∧ size M = card A}
  (is bij-betw - ?FSet ?MSet)
proof (rule bij-betw-byWitness[where f'=λM. functions-of A M])

```

```

have quotient-eq: ?FSet = {F ∈ ((A →E B) // domain-permutation A B). univ
(λf. f ‘ A = B) F}
using equiv-domain-permutation[of A B] surjective-respects-domain-permutation[of
A B]
by (simp only: univ-preserves-predicate)
show ∀f ∈ ?FSet. functions-of A (msubset-of A f) = f
using ⟨finite A⟩ by (auto simp only: quotient-eq functions-of-msubset-of)
show ∀M ∈ ?MSet. msubset-of A (functions-of A M) = M
using ⟨finite A⟩ msubset-of-functions-of by blast
show msubset-of A ‘ ?FSet ⊆ ?MSet
using ⟨finite A⟩ by (auto simp add: quotient-eq surj-on-implies-set-mset-eq
msubset-of)
show functions-of A ‘ ?MSet ⊆ ?FSet
using ⟨finite A⟩ by (auto simp add: quotient-eq intro: functions-of func-
tions-of-is-surj-on)
qed

```

8.3 Cardinality

lemma card-surjective-functions-domain-permutation:

```

assumes finite A finite B
assumes card B ≤ card A
shows card ({f ∈ A →E B. f ‘ A = B} // domain-permutation A B) = (card A
– 1) choose (card A – card B)
proof –
let ?FSet = {f ∈ A →E B. f ‘ A = B} // domain-permutation A B
and ?MSet = {M. set-mset M = B ∧ size M = card A}
have bij-betw (msubset-of A) ?FSet ?MSet
using ⟨finite A⟩ by (rule bij-betw-msubset-of)
from this have card ?FSet = card ?MSet
by (rule bij-betw-same-card)
also have card ?MSet = (card A – 1) choose (card A – card B)
using ⟨finite B⟩ ⟨card B ≤ card A⟩ by (rule card-multisets-covering-set)
finally show ?thesis .
qed

```

end

9 Functions from A to B up to a Permutation on B

```

theory Twelfefold-Way-Entry7
imports Equiv-Relations-on-Functions
begin

```

9.1 Definition of Bijections

```

definition partitions-of :: 'a set ⇒ 'b set ⇒ ('a ⇒ 'b) set ⇒ 'a set set
where

```

$partitions\text{-of } A B F = univ (\lambda f. (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}) F$

definition $functions\text{-of} :: 'a \text{ set set} \Rightarrow 'a \text{ set} \Rightarrow 'b \text{ set} \Rightarrow ('a \Rightarrow 'b) \text{ set}$
where

$functions\text{-of } P A B = \{f \in A \rightarrow_E B. (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\} = P\}$

9.2 Properties for Bijections

lemma $partitions\text{-of}$:

assumes $finite B$

assumes $F \in (A \rightarrow_E B) // \text{range-permutation } A B$

shows $card (partitions\text{-of } A B F) \leq card B$

and $partition\text{-on } A (partitions\text{-of } A B F)$

proof –

from $\langle F \in (A \rightarrow_E B) // \text{range-permutation } A B \rangle$ **obtain** f **where** $f \in A \rightarrow_E B$

and $F\text{-eq}$: $F = \text{range-permutation } A B \text{ “ } \{f\} \text{ using quotientE by blast}$

have $partitions\text{-of } A B F = univ (\lambda f. (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}) F$

unfolding $partitions\text{-of-def} ..$

also have $\dots = univ (\lambda f. (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}) (\text{range-permutation } A B \text{ “ } \{f\})$

unfolding $F\text{-eq} ..$

also have $\dots = (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}$

using $equiv\text{-range-permutation domain-partitions-respects-range-permutation} \langle f \in A \rightarrow_E B \rangle$

by $(subst univ\text{-commute}') auto$

finally have $partitions\text{-of-eq}$: $partitions\text{-of } A B F = (\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\} .$

show $card (partitions\text{-of } A B F) \leq card B$

proof –

have $card (partitions\text{-of } A B F) = card ((\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\})$

unfolding $partitions\text{-of-eq} ..$

also have $\dots \leq card ((\lambda b. \{x \in A. f x = b\}) ' B)$

using $\langle finite B \rangle$ **by** $(auto \text{ intro: } card\text{-mono})$

also have $\dots \leq card B$

using $\langle finite B \rangle$ **by** $(rule card\text{-image-le})$

finally show $?thesis .$

qed

show $partition\text{-on } A (partitions\text{-of } A B F)$

proof –

have $partition\text{-on } A ((\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\})$

using $\langle f \in A \rightarrow_E B \rangle$ **by** $(auto \text{ intro!}: partition\text{-onI})$

from this show $?thesis$

unfolding $partitions\text{-of-eq} .$

qed

qed

lemma $functions\text{-of}$:

assumes $finite A$ $finite B$

```

assumes partition-on  $A P$ 
assumes  $\text{card } P \leq \text{card } B$ 
shows functions-of  $P A B \in (A \rightarrow_E B) // \text{range-permutation } A B$ 
proof –
  obtain  $f$  where  $f \in A \rightarrow_E B$  and  $r1: (\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\} = P$ 
    using obtain-function-with-partition[ $OF \langle \text{finite } A \rangle \langle \text{finite } B \rangle \langle \text{partition-on } A P \rangle$ 
     $\langle \text{card } P \leq \text{card } B \rangle$ ]
    by blast
  have functions-of  $P A B = \text{range-permutation } A B \text{ ‘ ‘ } \{f\}$ 
  proof
    show functions-of  $P A B \subseteq \text{range-permutation } A B \text{ ‘ ‘ } \{f\}$ 
    proof
      fix  $f'$ 
      assume  $f' \in \text{functions-of } P A B$ 
      from this have  $f' \in A \rightarrow_E B$  and  $r2: (\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\}$ 
      =  $P$ 
      unfolding functions-of-def by auto
      from  $r1 r2$ 
      obtain  $p$  where  $p$  permutes  $B \wedge (\forall x \in A. f x = p (f' x))$ 
      using partitions-eq-implies-permutes[ $OF \langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle$ 
       $\langle \text{finite } B \rangle$ ] by metis
      from this show  $f' \in \text{range-permutation } A B \text{ ‘ ‘ } \{f\}$ 
      using  $\langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle$ 
      unfolding range-permutation-def by auto
    qed
  next
  show range-permutation  $A B \text{ ‘ ‘ } \{f\} \subseteq \text{functions-of } P A B$ 
  proof
    fix  $f'$ 
    assume  $f' \in \text{range-permutation } A B \text{ ‘ ‘ } \{f\}$ 
    from this have  $(f, f') \in \text{range-permutation } A B$  by auto
    from this have  $f' \in A \rightarrow_E B$ 
    unfolding range-permutation-def by auto
    from  $\langle (f, f') \in \text{range-permutation } A B \rangle$  have
       $(\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\}$ 
    using congruentD[ $OF \text{domain-partitions-respects-range-permutation}$ ] by blast
    from  $\langle f' \in A \rightarrow_E B \rangle$  this  $r1$  show  $f' \in \text{functions-of } P A B$ 
    unfolding functions-of-def by auto
  qed
qed
from this  $\langle f \in A \rightarrow_E B \rangle$  show ?thesis by (auto intro: quotientI)
qed

lemma functions-of-partitions-of:
assumes finite  $B$ 
assumes  $F \in (A \rightarrow_E B) // \text{range-permutation } A B$ 
shows functions-of (partitions-of  $A B F$ )  $A B = F$ 
proof –
  from  $\langle F \in (A \rightarrow_E B) // \text{range-permutation } A B \rangle$  obtain  $f$  where  $f \in A \rightarrow_E B$ 

```


B
and $F\text{-eq}$: $F = \text{range-permutation } A B \text{ “ } \{f\} \text{ using quotient } E \text{ by blast}$
have partitions-of-eq : $\text{partitions-of } A B F = (\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}$
unfolding $\text{partitions-of-def } F\text{-eq}$
using $\text{equiv-range-permutation domain-partitions-respects-range-permutation}$
 $\langle f \in A \rightarrow_E B \rangle$
by $(\text{subst univ-commute}^\wedge) \text{ auto}$
show $?thesis$
proof
show $\text{functions-of } (\text{partitions-of } A B F) A B \subseteq F$
proof
fix f'
assume f' : $f' \in \text{functions-of } (\text{partitions-of } A B F) A B$
from this **have** $(\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\}$
 $\text{‘ } B - \{\{\}\}$
unfolding functions-of-def **by** $(\text{auto simp add: partitions-of-eq})$
note $\langle f \in A \rightarrow_E B \rangle$
moreover from f' **have** $f' \in A \rightarrow_E B$
unfolding functions-of-def **by** auto
moreover obtain p **where** p $\text{permutes } B \wedge (\forall x \in A. f x = p (f' x))$
using $\text{partitions-eq-implies-permutes}[OF \langle f \in A \rightarrow_E B \rangle \langle f' \in A \rightarrow_E B \rangle]$
 $\langle \text{finite } B \rangle$
 $\langle (\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\} = (\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\} \rangle$
by metis
ultimately show $f' \in F$
unfolding $F\text{-eq range-permutation-def}$ **by** auto
qed
next
show $F \subseteq \text{functions-of } (\text{partitions-of } A B F) A B$
proof
fix f'
assume $f' \in F$
from this **have** $f' \in A \rightarrow_E B$
unfolding $F\text{-eq range-permutation-def}$ **by** auto
from $\langle f' \in F \rangle$ **obtain** p **where** p $\text{permutes } B \forall x \in A. f x = p (f' x)$
unfolding $F\text{-eq range-permutation-def}$ **by** auto
have eq : $(\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\} = (\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}$
proof –
have $(\lambda b. \{x \in A. f' x = b\}) \text{ ‘ } B - \{\{\}\} = (\lambda b. \{x \in A. p (f' x) = b\}) \text{ ‘ } B - \{\{\}\}$
using $\text{permutes-implies-inv-image-on-eq}[OF \langle p \text{ permutes } B \rangle, \text{ of } A f']$ **by**
 simp
also have $\dots = (\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}$
using $\langle \forall x \in A. f x = p (f' x) \rangle$ **by** auto
finally show $?thesis$.
qed
from this $\langle f' \in A \rightarrow_E B \rangle$ **show** $f' \in \text{functions-of } (\text{partitions-of } A B F) A B$
unfolding $\text{functions-of-def partitions-of-eq}$ **by** auto

qed
 qed
 qed

lemma *partitions-of-functions-of*:

assumes *finite A finite B*
assumes *partition-on A P*
assumes *card P ≤ card B*
shows *partitions-of A B (functions-of P A B) = P*

proof –

have *functions-of P A B ∈ (A →_E B) // range-permutation A B*
using *⟨finite A⟩ ⟨finite B⟩ ⟨partition-on A P⟩ ⟨card P ≤ card B⟩* **by** (*rule functions-of*)

from this obtain f where *f ∈ A →_E B* **and** *functions-of-eq: functions-of P A B = range-permutation A B “{f}”*

using *quotientE* **by** *metis*

from functions-of-eq *⟨f ∈ A →_E B⟩* **have** *f ∈ functions-of P A B*

using *equiv-range-permutation equiv-class-self* **by** *fastforce*

have *partitions-of A B (functions-of P A B) = univ (λf. (λb. {x ∈ A. f x = b}) ‘B - {{{}}’) (functions-of P A B)*

unfolding *partitions-of-def ..*

also have *... = univ (λf. (λb. {x ∈ A. f x = b}) ‘B - {{{}}’) (range-permutation A B “{f}”)*

unfolding *⟨functions-of P A B = range-permutation A B “{f}” ..*

also have *... = (λb. {x ∈ A. f x = b}) ‘B - {{{}}’*

using *equiv-range-permutation domain-partitions-respects-range-permutation* *⟨f ∈ A →_E B⟩*

by (*subst univ-commute’*) *auto*

also have *(λb. {x ∈ A. f x = b}) ‘B - {{{}}’ = P*

using *⟨f ∈ functions-of P A B⟩* **unfolding** *functions-of-def* **by** *simp*

finally show *?thesis .*

qed

9.3 Bijections

lemma *bij-betw-partitions-of*:

assumes *finite A finite B*

shows *bij-betw (partitions-of A B) ((A →_E B) // range-permutation A B) {P. partition-on A P ∧ card P ≤ card B}*

proof (*rule bij-betw-byWitness* **where** *f’ = λP. functions-of P A B*)

show *∀ F ∈ (A →_E B) // range-permutation A B. functions-of (partitions-of A B F) A B = F*

using *⟨finite B⟩* **by** (*simp add: functions-of-partitions-of*)

show *∀ P ∈ {P. partition-on A P ∧ card P ≤ card B}. partitions-of A B (functions-of P A B) = P*

using *⟨finite A⟩ ⟨finite B⟩* **by** (*auto simp add: partitions-of-functions-of*)

show *partitions-of A B ‘((A →_E B) // range-permutation A B) ⊆ {P. partition-on A P ∧ card P ≤ card B}*

using *⟨finite B⟩ partitions-of* **by** *auto*

```

  show (λP. functions-of P A B) ‘ {P. partition-on A P ∧ card P ≤ card B} ⊆
(A →E B) // range-permutation A B
  using functions-of ⟨finite A⟩ ⟨finite B⟩ by auto
qed

```

9.4 Cardinality

lemma

```

  assumes finite A finite B
  shows card ((A →E B) // range-permutation A B) = (∑ j≤card B. Stirling
(card A) j)
  proof –
  have bij-betw (partitions-of A B) ((A →E B) // range-permutation A B) {P.
partition-on A P ∧ card P ≤ card B}
  using ⟨finite A⟩ ⟨finite B⟩ by (rule bij-betw-partitions-of)
  from this have card ((A →E B) // range-permutation A B) = card {P. parti-
tion-on A P ∧ card P ≤ card B}
  by (rule bij-betw-same-card)
  also have card {P. partition-on A P ∧ card P ≤ card B} = (∑ j≤card B.
Stirling (card A) j)
  using ⟨finite A⟩ by (rule card-partition-on-at-most-size)
  finally show ?thesis .
qed

```

end

10 Injections from A to B up to a Permutation on B

```

theory Twelfefold-Way-Entry8
imports Twelfefold-Way-Entry7
begin

```

10.1 Properties for Bijections

lemma *inj-on-implies-partitions-of*:

```

  assumes F ∈ (A →E B) // range-permutation A B
  assumes univ (λf. inj-on f A) F
  shows ∀ X ∈ partitions-of A B F. card X = 1
  proof –
  from ⟨F ∈ (A →E B) // range-permutation A B⟩ obtain f where f ∈ A →E
B
  and F-eq: F = range-permutation A B “ {f} using quotientE by blast
  from this ⟨univ (λf. inj-on f A) F⟩ have inj-on f A
  using univ-commute'[OF equiv-range-permutation inj-on-respects-range-permutation
⟨f ∈ A →E B⟩] by simp
  have ∀ X∈(λb. {x ∈ A. f x = b}) ‘ B - {{{}}. card X = 1
  proof
  fix X

```

```

assume  $X \in (\lambda b. \{x \in A. f x = b\}) \text{ ` } B - \{\{\}\}$ 
from this obtain  $x$  where  $X = \{xa \in A. f xa = f x\}$   $x \in A$  by auto
from this have  $X = \{x\}$ 
  using  $\langle inj\text{-on } f A \rangle$  by  $(auto\ dest!\!: inj\text{-on}D)$ 
from this show  $card\ X = 1$  by simp
qed
from this show ?thesis
  unfolding partitions-of-def F-eq
  using equiv-range-permutation domain-partitions-respects-range-permutation  $\langle f$ 
 $\in A \rightarrow_E B \rangle$ 
  by  $(subst\ univ\text{-commute}')\ assumption+$ 
qed

```

lemma *unique-part-eq-singleton*:

```

assumes partition-on A P
assumes  $\forall X \in P. card\ X = 1$ 
assumes  $x \in A$ 
shows  $(THE\ X. x \in X \wedge X \in P) = \{x\}$ 
proof –
  have  $(THE\ X. x \in X \wedge X \in P) \in P$ 
    using  $\langle partition\text{-on } A P \rangle \langle x \in A \rangle$  by  $(simp\ add:\ partition\text{-on}\text{-the}\text{-part}\text{-mem})$ 
  from this have  $card\ (THE\ X. x \in X \wedge X \in P) = 1$ 
    using  $\langle \forall X \in P. card\ X = 1 \rangle$  by auto
  moreover have  $x \in (THE\ X. x \in X \wedge X \in P)$ 
    using  $\langle partition\text{-on } A P \rangle \langle x \in A \rangle$  by  $(simp\ add:\ partition\text{-on}\text{-in}\text{-the}\text{-unique}\text{-part})$ 
  ultimately show ?thesis
    by  $(metis\ card\text{-}1\text{-singleton}E\ singleton\text{-iff})$ 
qed

```

lemma *functions-of-is-inj-on*:

```

assumes finite A finite B partition-on A P card P ≤ card B
assumes  $\forall X \in P. card\ X = 1$ 
shows univ  $(\lambda f. inj\text{-on } f A)$   $(functions\text{-of } P A B)$ 
proof –
  have functions-of P A B  $\in (A \rightarrow_E B) // range\text{-permutation } A B$ 
    using functions-of  $\langle finite\ A \rangle \langle finite\ B \rangle \langle partition\text{-on } A P \rangle \langle card\ P \leq card\ B \rangle$ 
by blast
  from this obtain  $f$  where eq-f: functions-of P A B = range-permutation A B
  “ $\{f\}$  and  $f \in A \rightarrow_E B$ 
    using quotientE by blast
  from eq-f have  $f \in functions\text{-of } P A B$ 
    using  $\langle f \in A \rightarrow_E B \rangle$  equiv-range-permutation equiv-class-self by fastforce
  from this have eq:  $(\lambda b. \{x \in A. f x = b\}) \text{ ` } B - \{\{\}\} = P$ 
    unfolding functions-of-def by auto
  have inj-on f A
  proof  $(rule\ inj\text{-on}I)$ 
    fix  $x\ y$ 
    assume  $x \in A\ y \in A\ f x = f y$ 
    from  $\langle x \in A \rangle$  have  $x \in \{x' \in A. f x' = f x\}$  by auto

```

moreover from $\langle y \in A \rangle \langle f x = f y \rangle$ **have** $y \in \{x' \in A. f x' = f x\}$ **by** *auto*
moreover have $\text{card } \{x' \in A. f x' = f x\} = 1$
proof –
from $\langle x \in A \rangle \langle f \in A \rightarrow_E B \rangle$ **have** $f x \in B$ **by** *auto*
from this $\langle x \in A \rangle$ **have** $\{x' \in A. f x' = f x\} \in (\lambda b. \{x \in A. f x = b\}) \text{ ` } B -$
 $\{\{\}\}$ **by** *auto*
from this $\langle \forall X \in P. \text{card } X = 1 \rangle$ **eq show** *?thesis* **by** *auto*
qed
ultimately show $x = y$ **by** (*metis card-1-singletonE singletonD*)
qed
from this show *?thesis*
unfolding *eq-f using equiv-range-permutation inj-on-respects-range-permutation*
 $\langle f \in A \rightarrow_E B \rangle$
by (*subst univ-commute'*) *assumption+*
qed

10.2 Bijections

lemma *bij-betw-partitions-of:*

assumes *finite A finite B*
shows *bij-betw (partitions-of A B) ({f \in A \to_E B. inj-on f A} // range-permutation A B) {P. partition-on A P \wedge card P \leq card B \wedge (\forall X \in P. card X = 1)}*
proof (*rule bij-betw-byWitness* **where** $f' = \lambda P. \text{functions-of } P \text{ A B}$)
have *quotient-eq: {f \in A \to_E B. inj-on f A} // range-permutation A B = {F \in ((A \to_E B) // range-permutation A B). univ (\lambda f. inj-on f A) F}*
by (*simp add: equiv-range-permutation inj-on-respects-range-permutation univ-preserves-predicate*)
show $\forall F \in \{f \in A \rightarrow_E B. \text{inj-on } f \text{ A}\} // \text{range-permutation } A \text{ B. functions-of (partitions-of } A \text{ B } F) \text{ A B} = F$
using $\langle \text{finite } B \rangle$ **by** (*simp add: quotient-eq functions-of-partitions-of*)
show $\forall P \in \{P. \text{partition-on } A \text{ P} \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\}$.
 $\text{partitions-of } A \text{ B (functions-of } P \text{ A B)} = P$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*simp add: partitions-of-functions-of*)
show *partitions-of A B ' ({f \in A \to_E B. inj-on f A} // range-permutation A B)*
 $\subseteq \{P. \text{partition-on } A \text{ P} \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\}$
using $\langle \text{finite } B \rangle$ *quotient-eq partitions-of inj-on-implies-partitions-of* **by** *fastforce*
show $(\lambda P. \text{functions-of } P \text{ A B}) \text{ ` } \{P. \text{partition-on } A \text{ P} \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\} \subseteq \{f \in A \rightarrow_E B. \text{inj-on } f \text{ A}\} // \text{range-permutation } A \text{ B}$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*auto simp add: quotient-eq intro: functions-of functions-of-is-inj-on*)
qed

10.3 Cardinality

lemma *card-injective-functions-range-permutation:*

assumes *finite A finite B*
shows $\text{card } (\{f \in A \rightarrow_E B. \text{inj-on } f \text{ A}\} // \text{range-permutation } A \text{ B}) = \text{iverson} (\text{card } A \leq \text{card } B)$
proof –
obtain *enum* **where** *bij-betw enum {0..<card A} A*
using $\langle \text{finite } A \rangle$ *ex-bij-betw-nat-finite* **by** *blast*

```

have bij-betw (partitions-of  $A\ B$ ) ( $\{f \in A \rightarrow_E B. \text{inj-on } f\ A\}$  // range-permutation
 $A\ B$ )  $\{P. \text{partition-on } A\ P \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\}$ 
  using  $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$  by (rule bij-betw-partitions-of)
  from this have card ( $\{f \in A \rightarrow_E B. \text{inj-on } f\ A\}$  // range-permutation  $A\ B$ ) =
card  $\{P. \text{partition-on } A\ P \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\}$ 
  by (rule bij-betw-same-card)
  also have card  $\{P. \text{partition-on } A\ P \wedge \text{card } P \leq \text{card } B \wedge (\forall X \in P. \text{card } X = 1)\}$  =
iverson ( $\text{card } A \leq \text{card } B$ )
  using  $\langle \text{finite } A \rangle$  by (rule card-partition-on-size1-eq-iverson)
  finally show ?thesis .
qed

end

```

11 Surjections from A to B up to a Permutation on B

```

theory Twelvefold-Way-Entry9
imports Twelvefold-Way-Entry7
begin

```

11.1 Properties for Bijections

```

lemma surjective-on-implies-card-eq:
  assumes  $f \text{ ' } A = B$ 
  shows card  $((\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\}) = \text{card } B$ 
proof -
  from  $\langle f \text{ ' } A = B \rangle$  have  $\{\} \notin (\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B$  by auto
  from  $\langle f \text{ ' } A = B \rangle$  have inj-on  $(\lambda b. \{x \in A. f\ x = b\})\ B$  by (fastforce intro:
inj-onI)
  have card  $((\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\}) = \text{card } ((\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B)$ 
  using  $\langle \{\} \notin (\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B \rangle$  by simp
  also have  $\dots = \text{card } B$ 
  using  $\langle \text{inj-on } (\lambda b. \{x \in A. f\ x = b\})\ B \rangle$  by (rule card-image)
  finally show ?thesis .
qed

```

```

lemma card-eq-implies-surjective-on:
  assumes finite  $B\ f \in A \rightarrow_E B$ 
  assumes card-eq: card  $((\lambda b. \{x \in A. f\ x = b\}) \text{ ' } B - \{\{\}\}) = \text{card } B$ 
  shows  $f \text{ ' } A = B$ 
proof
  from  $\langle f \in A \rightarrow_E B \rangle$  show  $f \text{ ' } A \subseteq B$  by auto
next
  show  $B \subseteq f \text{ ' } A$ 
  proof
    fix  $x$ 

```

```

assume  $x \in B$ 
have  $\{\} \notin (\lambda b. \{x \in A. f x = b\}) \text{ ' } B$ 
proof (cases card  $B \geq 1$ )
  assume  $\neg \text{card } B \geq 1$ 
  from this have card  $B = 0$  by simp
  from this  $\langle \text{finite } B \rangle$  have  $B = \{\}$  by simp
  from this show ?thesis by simp
next
assume card  $B \geq 1$ 
show ?thesis
proof (rule ccontr)
  assume  $\neg \{\} \notin (\lambda b. \{x \in A. f x = b\}) \text{ ' } B$ 
  from this have  $\{\} \in (\lambda b. \{x \in A. f x = b\}) \text{ ' } B$  by simp
  moreover have card  $((\lambda b. \{x \in A. f x = b\}) \text{ ' } B) \leq \text{card } B$ 
    using  $\langle \text{finite } B \rangle$  card-image-le by blast
  moreover have finite  $((\lambda b. \{x \in A. f x = b\}) \text{ ' } B)$ 
    using  $\langle \text{finite } B \rangle$  by auto
  ultimately have card  $((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) \leq \text{card } B - 1$ 
    by (auto simp add: card-Diff-singleton)
  from this card-eq  $\langle \text{card } B \geq 1 \rangle$  show False by auto
qed
qed
from this  $\langle x \in B \rangle$  show  $x \in f \text{ ' } A$  by force
qed
qed

lemma card-partitions-of:
  assumes  $F \in (A \rightarrow_E B)$  // range-permutation  $A B$ 
  assumes univ  $(\lambda f. f \text{ ' } A = B) F$ 
  shows card (partitions-of  $A B F$ ) = card  $B$ 
proof -
from  $\langle F \in (A \rightarrow_E B)$  // range-permutation  $A B \rangle$  obtain  $f$  where  $f \in A \rightarrow_E B$ 
  and  $F$ -eq:  $F = \text{range-permutation } A B \text{ " } \{f\}$  using quotientE by blast
  from this  $\langle \text{univ } (\lambda f. f \text{ ' } A = B) F \rangle$  have  $f \text{ ' } A = B$ 
  using univ-commute'[OF equiv-range-permutation surj-on-respects-range-permutation
 $\langle f \in A \rightarrow_E B \rangle$ ] by simp
  have card (partitions-of  $A B F$ ) = card (univ  $(\lambda f. (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) F$ )
  unfolding partitions-of-def ..
  also have  $\dots = \text{card } (\text{univ } (\lambda f. (\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\}) (\text{range-permutation } A B \text{ " } \{f\}))$ 
  unfolding  $F$ -eq ..
  also have  $\dots = \text{card } ((\lambda b. \{x \in A. f x = b\}) \text{ ' } B - \{\{\}\})$ 
  using equiv-range-permutation domain-partitions-respects-range-permutation  $\langle f \in A \rightarrow_E B \rangle$ 
  by (subst univ-commute') auto
  also from  $\langle f \text{ ' } A = B \rangle$  have  $\dots = \text{card } B$ 
  using surjective-on-implies-card-eq by auto

```

finally show *?thesis* .
qed

lemma *functions-of-is-surj-on*:

assumes *finite A finite B*
assumes *partition-on A P card P = card B*
shows *univ (λf. f ‘ A = B) (functions-of P A B)*

proof –

have *functions-of P A B ∈ (A →_E B) // range-permutation A B*
using *functions-of ⟨finite A⟩ ⟨finite B⟩ ⟨partition-on A P⟩ ⟨card P = card B⟩*
by *fastforce*
from *this* **obtain** *f* **where** *eq-f: functions-of P A B = range-permutation A B*
“*{f}*” **and** *f ∈ A →_E B*
using *quotientE* **by** *blast*
from *eq-f* **have** *f ∈ functions-of P A B*
using *⟨f ∈ A →_E B⟩ equiv-range-permutation equiv-class-self* **by** *fastforce*
from *⟨f ∈ functions-of P A B⟩* **have** *eq: (λb. {x ∈ A. f x = b}) ‘ B – {{{}} = P*
unfolding *functions-of-def* **by** *auto*
from *this* **have** *card ((λb. {x ∈ A. f x = b}) ‘ B – {{{}}) = card B*
using *⟨card P = card B⟩* **by** *simp*
from *⟨finite B⟩ ⟨f ∈ A →_E B⟩* **this** **have** *f ‘ A = B*
using *card-eq-implies-surjective-on* **by** *blast*
from *this* **show** *?thesis*
unfolding *eq-f* **using** *equiv-range-permutation surj-on-respects-range-permutation*
⟨f ∈ A →_E B⟩
by *(subst univ-commute’)* *assumption+*
qed

11.2 Bijections

lemma *bij-betw-partitions-of*:

assumes *finite A finite B*
shows *bij-betw (partitions-of A B) ({f ∈ A →_E B. f ‘ A = B} // range-permutation A B) {P. partition-on A P ∧ card P = card B}*
proof (*rule bij-betw-byWitness* **where** *f’=λP. functions-of P A B*)
have *quotient-eq: {f ∈ A →_E B. f ‘ A = B} // range-permutation A B = {F ∈ ((A →_E B) // range-permutation A B). univ (λf. f ‘ A = B) F}*
using *equiv-range-permutation[of A B] surj-on-respects-range-permutation[of A B]* **by** (*simp only: univ-preserves-predicate*)
show *∀ F ∈ {f ∈ A →_E B. f ‘ A = B} // range-permutation A B. functions-of (partitions-of A B F) A B = F*
using *⟨finite B⟩* **by** (*simp add: functions-of-partitions-of quotient-eq*)
show *∀ P ∈ {P. partition-on A P ∧ card P = card B}. partitions-of A B (functions-of P A B) = P*
using *⟨finite A⟩ ⟨finite B⟩* **by** (*auto simp add: partitions-of-functions-of*)
show *partitions-of A B ‘ ({f ∈ A →_E B. f ‘ A = B} // range-permutation A B)*
 \subseteq *{P. partition-on A P ∧ card P = card B}*
using *⟨finite B⟩ quotient-eq card-partitions-of partitions-of* **by** *fastforce*
show *(λP. functions-of P A B) ‘ {P. partition-on A P ∧ card P = card B} ⊆*


```

{f ∈ A →E B. f ‘ A = B} // range-permutation A B
  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp add: quotient-eq intro: functions-of
functions-of-is-surj-on)
qed

```

11.3 Cardinality

lemma *card-surjective-functions-range-permutation:*

```

  assumes finite A finite B
  shows card ({f ∈ A →E B. f ‘ A = B} // range-permutation A B) = Stirling
(card A) (card B)
proof –
  have bij-betw (partitions-of A B) ({f ∈ A →E B. f ‘ A = B} // range-permutation
A B) {P. partition-on A P ∧ card P = card B}
    using ⟨finite A⟩ ⟨finite B⟩ by (rule bij-betw-partitions-of)
  from this have card ({f ∈ A →E B. f ‘ A = B} // range-permutation A B) =
card {P. partition-on A P ∧ card P = card B}
    by (rule bij-betw-same-card)
  also have card {P. partition-on A P ∧ card P = card B} = Stirling (card A)
(card B)
    using ⟨finite A⟩ by (rule card-partition-on)
  finally show ?thesis .
qed

```

end

12 Surjections from A to B

theory *Twelvefold-Way-Entry3*

imports

Twelvefold-Way-Entry9

begin

lemma *card-of-equiv-class:*

```

  assumes finite B
  assumes F ∈ {f ∈ A →E B. f ‘ A = B} // range-permutation A B
  shows card F = fact (card B)
proof –
  from ⟨F ∈ {f ∈ A →E B. f ‘ A = B} // range-permutation A B⟩ obtain f
where
    f ∈ A →E B and f ‘ A = B
    and F-eq: F = range-permutation A B “ {f} using quotientE by blast
  have set-eq: range-permutation A B “ {f} = (λp x. if x ∈ A then p (f x) else
undefined) ‘ {p. p permutes B}
    proof
  show range-permutation A B “ {f} ⊆ (λp x. if x ∈ A then p (f x) else undefined)
‘ {p. p permutes B}
    proof
    fix f'

```

```

assume  $f' \in \text{range-permutation } A \ B \ \{\!| \ f \ |\}$ 
from this obtain  $p$  where  $p$  permutes  $B \ \forall x \in A. f \ x = p \ (f' \ x)$ 
  unfolding range-permutation-def by auto
from  $\langle f' \in \text{range-permutation } A \ B \ \{\!| \ f \ |\} \rangle$  have  $f' \in A \rightarrow_E \ B$ 
  unfolding range-permutation-def by auto
have  $f' = (\lambda x. \text{if } x \in A \text{ then } \text{inv } p \ (f \ x) \text{ else } \text{undefined})$ 
proof
  fix  $x$ 
  show  $f' \ x = (\text{if } x \in A \text{ then } \text{inv } p \ (f \ x) \text{ else } \text{undefined})$ 
    using  $\langle f' \in A \rightarrow_E \ B \rangle \langle f' \in A \rightarrow_E \ B \rangle \langle \forall x \in A. f \ x = p \ (f' \ x) \rangle$ 
     $\langle p \text{ permutes } B \rangle$  permutes-inverses(2) by fastforce
qed
  moreover have  $\text{inv } p$  permutes  $B$  using  $\langle p \text{ permutes } B \rangle$  by (simp add:
permutes-inv)
  ultimately show  $f' \in (\lambda p. (\lambda x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined})) \ \{\!| \ p. \ p \text{ permutes } B \ |\}$ 
    by auto
qed
next
  show  $(\lambda p \ x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined}) \ \{\!| \ p. \ p \text{ permutes } B \ |\} \subseteq$ 
range-permutation  $A \ B \ \{\!| \ f \ |\}$ 
proof
  fix  $f'$ 
  assume  $f' \in (\lambda p \ x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined}) \ \{\!| \ p. \ p \text{ permutes } B \ |\}$ 
  from this obtain  $p$  where  $p$  permutes  $B$  and  $f'$ -eq:  $f' = (\lambda x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined})$  by auto
  from this have  $f' \in A \rightarrow_E \ B$ 
    using  $\langle f' \in A \rightarrow_E \ B \rangle$  permutes-in-image by fastforce
    moreover have  $\text{inv } p$  permutes  $B$  using  $\langle p \text{ permutes } B \rangle$  by (simp add:
permutes-inv)
    moreover have  $\forall x \in A. f \ x = \text{inv } p \ (f' \ x)$ 
      using  $\langle f' \in A \rightarrow_E \ B \rangle \langle f' \in A \rightarrow_E \ B \rangle$   $f'$ -eq
       $\langle p \text{ permutes } B \rangle$  permutes-inverses(2) by fastforce
    ultimately show  $f' \in \text{range-permutation } A \ B \ \{\!| \ f \ |\}$ 
      using  $\langle f' \in A \rightarrow_E \ B \rangle$  unfolding range-permutation-def by auto
qed
qed
have inj-on  $(\lambda p \ x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined}) \ \{\!| \ p. \ p \text{ permutes } B \ |\}$ 
proof (rule inj-onI)
  fix  $p \ p'$ 
  assume  $p \in \{\!| \ p. \ p \text{ permutes } B \ |\} \ p' \in \{\!| \ p. \ p \text{ permutes } B \ |\}$ 
  and eq:  $(\lambda x. \text{if } x \in A \text{ then } p \ (f \ x) \text{ else } \text{undefined}) = (\lambda x. \text{if } x \in A \text{ then } p' \ (f \ x) \text{ else } \text{undefined})$ 
  {
    fix  $x$ 
    have  $p \ x = p' \ x$ 
    proof cases
      assume  $x \in B$ 
      from this obtain  $y$  where  $y \in A$  and  $x = f \ y$ 

```

```

    using ⟨f ‘ A = B⟩ by blast
    from eq this have p (f y) = p' (f y) by meson
    from this ⟨x = f y⟩ show p x = p' x by simp
next
  assume x ∉ B
  from this show p x = p' x
    using ⟨p ∈ {p. p permutes B}⟩ ⟨p' ∈ {p. p permutes B}⟩
    by (simp add: permutes-def)
qed
}
from this show p = p' by auto
qed
have card F = card ((λp x. if x ∈ A then p (f x) else undefined) ‘ {p. p permutes
B})
  unfolding F-eq set-eq ..
  also have ... = card {p. p permutes B}
    using ⟨inj-on (λp x. if x ∈ A then p (f x) else undefined) {p. p permutes B}⟩
    by (simp add: card-image)
  also have ... = fact (card B)
    using ⟨finite B⟩ by (simp add: card-permutations)
  finally show ?thesis .
qed

```

```

lemma card-extensional-funcset-surj-on:
  assumes finite A finite B
  shows card {f ∈ A →E B. f ‘ A = B} = fact (card B) * Stirling (card A) (card
B) (is card ?F = -)
proof -
  have card ?F = fact (card B) * card (?F // range-permutation A B)
    using ⟨finite B⟩
    by (simp only: card-equiv-class-restricted-same-size[OF equiv-range-permutation
surj-on-respects-range-permutation card-of-equiv-class])
  also have ... = fact (card B) * Stirling (card A) (card B)
    using ⟨finite A⟩ ⟨finite B⟩
    by (simp only: card-surjective-functions-range-permutation)
  finally show ?thesis .
qed
end

```

13 Functions from A to B up to a Permutation on A and B

```

theory Twelvefold-Way-Entry10
imports Equiv-Relations-on-Functions
begin

```

13.1 Definition of Bijections

definition *number-partition-of* :: 'a set \Rightarrow 'b set \Rightarrow ('a \Rightarrow 'b) set \Rightarrow nat multiset
where

number-partition-of A B F = univ (λf . image-mset (λX . card X) (mset-set ((λb . { $x \in A$. f x = b}) ' B - {{{}}})) F

definition *functions-of* :: 'a set \Rightarrow 'b set \Rightarrow nat multiset \Rightarrow ('a \Rightarrow 'b) set
where

functions-of A B N = { $f \in A \rightarrow_E B$. image-mset (λX . card X) (mset-set ((λb . { $x \in A$. f x = b}) ' B - {{{}}})) = N}

13.2 Properties for Bijections

lemma *card-setsum-partition*:

assumes *finite* A *finite* B $f \in A \rightarrow_E B$

shows *sum card* ((λb . { $x \in A$. f x = b}) ' B - {{{}}}) = *card* A

proof –

have *finite* ((λb . { $x \in A$. f x = b}) ' B - {{{}}})

using <*finite* B> **by** *blast*

moreover have $\forall X \in (\lambda b$. { $x \in A$. f x = b}) ' B - {{{}}}. *finite* X

using <*finite* A> **by** *auto*

moreover have $\bigcup ((\lambda b$. { $x \in A$. f x = b}) ' B - {{{}}}) = A

using < $f \in A \rightarrow_E B$ > **by** *auto*

ultimately show *?thesis*

by (*subst card-Union-disjoint[symmetric]*) (*auto simp: pairwise-def disjnt-def*)

qed

lemma *number-partition-of*:

assumes *finite* A *finite* B

assumes $F \in (A \rightarrow_E B)$ // *domain-and-range-permutation* A B

shows *number-partition* (*card* A) (*number-partition-of* A B F)

and *size* (*number-partition-of* A B F) \leq *card* B

proof –

from < $F \in (A \rightarrow_E B)$ // *domain-and-range-permutation* A B> **obtain** *f* **where**
 $f \in A \rightarrow_E B$

and *F-eq*: $F = \text{domain-and-range-permutation } A \ B \ \text{"\{f\}}$ **using** *quotientE* **by**
blast

have *number-partition-of-eq*: *number-partition-of* A B F = *image-mset card*
(*mset-set* ((λb . { $x \in A$. f x = b}) ' B - {{{}}}))

proof –

have *number-partition-of* A B F = univ (λf . *image-mset card* (*mset-set* ((λb .
{ $x \in A$. f x = b}) ' B - {{{}}})) F

unfolding *number-partition-of-def* ..

also have ... = univ (λf . *image-mset card* (*mset-set* ((λb . { $x \in A$. f x = b}) ' B - {{{}}})) (*domain-and-range-permutation* A B " {f}))

unfolding *F-eq* ..

also have ... = *image-mset card* (*mset-set* ((λb . { $x \in A$. f x = b}) ' B - {{{}}}))

using <*finite* B> *equiv-domain-and-range-permutation multiset-of-partition-cards-respects-domain-and-range*

```

⟨f ∈ A →E B⟩
  by (subst univ-commute') auto
  finally show ?thesis .
qed
show number-partition (card A) (number-partition-of A B F)
proof -
  have sum-mset (number-partition-of A B F) = card A
    using number-partition-of-eq ⟨finite A⟩ ⟨finite B⟩ ⟨f ∈ A →E B⟩
    by (simp only: sum-unfold-sum-mset[symmetric] card-setsum-partition)
  moreover have 0 ∉# number-partition-of A B F
  proof -
    have ∀X ∈ (λb. {x ∈ A. f x = b}) ‘ B. finite X
      using ⟨finite A⟩ by simp
    from this have ∀X ∈ (λb. {x ∈ A. f x = b}) ‘ B - {{}}. card X ≠ 0 by
auto
    from this show ?thesis
      using number-partition-of-eq ⟨finite B⟩ by (simp add: image-iff)
  qed
  ultimately show ?thesis unfolding number-partition-def by simp
qed
show size (number-partition-of A B F) ≤ card B
  using number-partition-of-eq ⟨finite A⟩ ⟨finite B⟩
  by (metis (no-types, lifting) card-Diff1-le card-image-le finite-imageI le-trans
size-image-mset size-mset-set)
qed

lemma functions-of:
  assumes finite A finite B
  assumes number-partition (card A) N
  assumes size N ≤ card B
  shows functions-of A B N ∈ (A →E B) // domain-and-range-permutation A B
proof -
  obtain f where f ∈ A →E B and eq-N: image-mset (λX. card X) (mset-set
(((λb. {x ∈ A. f x = b}))) ‘ B - {{}})) = N
    using obtain-extensional-function-from-number-partition ⟨finite A⟩ ⟨finite B⟩
⟨number-partition (card A) N⟩ ⟨size N ≤ card B⟩ by blast
  have functions-of A B N = (domain-and-range-permutation A B) “ {f}
  proof
    show functions-of A B N ⊆ domain-and-range-permutation A B “ {f}
  proof
    fix f'
    assume f' ∈ functions-of A B N
    from this have eq-N': N = image-mset (λX. card X) (mset-set (((λb. {x ∈
A. f' x = b}))) ‘ B - {{}}))
      and f' ∈ A →E B
    unfolding functions-of-def by auto
    from ⟨finite A⟩ ⟨finite B⟩ ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩
    obtain pA pB where pA permutes A pB permutes B ∀x ∈ A. f x = pB (f' (pA
x))

```

```

    using eq-N eq-N' multiset-of-partition-cards-eq-implies-permutes[of A B f f']
  by blast
  from this show f' ∈ domain-and-range-permutation A B “ {f}
    using ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩
    unfolding domain-and-range-permutation-def by auto
  qed
next
show domain-and-range-permutation A B “ {f} ⊆ functions-of A B N
proof
  fix f'
  assume f' ∈ domain-and-range-permutation A B “ {f}
  from this have in-equiv-relation: (f, f') ∈ domain-and-range-permutation A
B by auto
  from eq-N ⟨finite B⟩ have image-mset (λX. card X) (mset-set (((λb. {x ∈
A. f' x = b})) ‘ B - {f}))) = N
  using congruentD[OF multiset-of-partition-cards-respects-domain-and-range-permutation
in-equiv-relation]
  by metis
  moreover from ⟨(f, f') ∈ domain-and-range-permutation A B⟩ have f' ∈ A
→E B
  unfolding domain-and-range-permutation-def by auto
  ultimately show f' ∈ functions-of A B N
  unfolding functions-of-def by auto
qed
qed
from this ⟨f ∈ A →E B⟩ show ?thesis by (auto intro: quotientI)
qed

```

lemma *functions-of-number-partition-of:*

```

  assumes finite A finite B
  assumes F ∈ (A →E B) // domain-and-range-permutation A B
  shows functions-of A B (number-partition-of A B F) = F
proof -
  from ⟨F ∈ (A →E B) // domain-and-range-permutation A B⟩ obtain f where
f ∈ A →E B
  and F-eq: F = domain-and-range-permutation A B “ {f} using quotientE by
blast
  have number-partition-of A B F = univ (λf. image-mset card (mset-set ((λb. {x
∈ A. f x = b}) ‘ B - {f}))) F
  unfolding number-partition-of-def ..
  also have ... = univ (λf. image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘
B - {f}))) (domain-and-range-permutation A B “ {f})
  unfolding F-eq ..
  also have ... = image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘ B - {f}))
  using ⟨finite B⟩
  using equiv-domain-and-range-permutation multiset-of-partition-cards-respects-domain-and-range-permutati
⟨f ∈ A →E B⟩
  by (subst univ-commute') auto
  finally have number-partition-of-eq: number-partition-of A B F = image-mset

```

```

card (mset-set ((λb. {x ∈ A. f x = b}) ‘ B - {{{}})) .
show ?thesis
proof
  show functions-of A B (number-partition-of A B F) ⊆ F
  proof
    fix f'
    assume f' ∈ functions-of A B (number-partition-of A B F)
    from this have f' ∈ A →E B
    and eq: image-mset card (mset-set ((λb. {x ∈ A. f' x = b}) ‘ B - {{{}}))
= image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘ B - {{{}}))
    unfolding functions-of-def by (auto simp add: number-partition-of-eq)
    note ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩
    moreover obtain pA pB where pA permutes A pB permutes B ∀x∈A. f x
= pB (f' (pA x))
    using ⟨finite A⟩ ⟨finite B⟩ ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩ eq
    multiset-of-partition-cards-eq-implies-permutes[of A B f f']
    by metis
    ultimately show f' ∈ F
    unfolding F-eq domain-and-range-permutation-def by auto
  qed
next
show F ⊆ functions-of A B (number-partition-of A B F)
proof
  fix f'
  assume f' ∈ F
  from ⟨f' ∈ F⟩ obtain pA pB where pA permutes A pB permutes B ∀x∈A.
f x = pB (f' (pA x))
  unfolding F-eq domain-and-range-permutation-def by auto
  have eq: image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘ B - {{{}})) =
image-mset card (mset-set ((λb. {x ∈ A. f' x = b}) ‘ B - {{{}}))
  proof -
    have (λb. {x ∈ A. f x = b}) ‘ B = (λb. {x ∈ A. pB (f' (pA x)) = b}) ‘ B
    using ⟨∀x∈A. f x = pB (f' (pA x))⟩ by auto
    from this have image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘ B -
{{{}})) =
image-mset card (mset-set ((λb. {x ∈ A. pB (f' (pA x)) = b}) ‘ B - {{{}}))
  by simp
  also have ... = image-mset card (mset-set ((λb. {x ∈ A. f' x = b}) ‘ B -
{{{}}))
  using ⟨pA permutes A⟩ ⟨pB permutes B⟩ permutes-implies-multiset-of-partition-cards-eq
  by blast
  finally show ?thesis .
  qed
  moreover from ⟨f' ∈ F⟩ have f' ∈ A →E B
  unfolding F-eq domain-and-range-permutation-def by auto
  ultimately show f' ∈ functions-of A B (number-partition-of A B F)
  unfolding functions-of-def number-partition-of-eq by auto
  qed
qed

```

qed

lemma *number-partition-of-functions-of:*

assumes *finite A finite B*

assumes *number-partition (card A) N size N ≤ card B*

shows *number-partition-of A B (functions-of A B N) = N*

proof –

from *assms have functions-of A B N ∈ (A →_E B) // domain-and-range-permutation A B*

using *functions-of assms by fastforce*

from this obtain f where *f ∈ A →_E B and functions-of A B N = domain-and-range-permutation A B “{f}”*

by *(meson quotientE)*

from this have *f ∈ functions-of A B N*

using *equiv-domain-and-range-permutation equiv-class-self by fastforce*

have *number-partition-of A B (functions-of A B N) = univ (λf. image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘B - {{{}}})) (functions-of A B N)*

unfolding *number-partition-of-def ..*

also have *... = univ (λf. image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘B - {{{}}})) (domain-and-range-permutation A B “{f}”)*

unfolding *{functions-of A B N = domain-and-range-permutation A B “{f}”*

..

also have *... = image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘B - {{{}}})*

using *{finite B} {f ∈ A →_E B} equiv-domain-and-range-permutation multiset-of-partition-cards-respects-domain-and-range-permutation*

by *(subst univ-commute’) auto*

also have *image-mset card (mset-set ((λb. {x ∈ A. f x = b}) ‘B - {{{}}}) = N*

using *{f ∈ functions-of A B N} unfolding functions-of-def by simp*

finally show *?thesis .*

qed

13.3 Bijections

lemma *bij-betw-number-partition-of:*

assumes *finite A finite B*

shows *bij-betw (number-partition-of A B) ((A →_E B) // domain-and-range-permutation A B) {N. number-partition (card A) N ∧ size N ≤ card B}*

proof *(rule bij-betw-byWitness[where f’=λM. functions-of A B M])*

show *∀ F ∈ (A →_E B) // domain-and-range-permutation A B. functions-of A B (number-partition-of A B F) = F*

using *{finite A} {finite B} by (auto simp add: functions-of-number-partition-of)*

show *∀ N ∈ {N. number-partition (card A) N ∧ size N ≤ card B}. number-partition-of A B (functions-of A B N) = N*

using *{finite A} {finite B} by (auto simp add: number-partition-of-functions-of)*

show *number-partition-of A B ‘ ((A →_E B) // domain-and-range-permutation A B) ⊆ {N. number-partition (card A) N ∧ size N ≤ card B}*

using *number-partition-of[of A B] {finite A} {finite B} by auto*

show *functions-of A B ‘ {N. number-partition (card A) N ∧ size N ≤ card B} ⊆ (A →_E B) // domain-and-range-permutation A B*

using *functions-of* \langle finite A \rangle \langle finite B \rangle by *blast*
qed

13.4 Cardinality

lemma *card-domain-and-range-permutation*:

assumes *finite A finite B*

shows $\text{card } ((A \rightarrow_E B) // \text{domain-and-range-permutation } A B) = \text{Partition } (\text{card } A + \text{card } B) (\text{card } B)$

proof –

have *bij-betw (number-partition-of A B) ((A →_E B) // domain-and-range-permutation A B) {N. number-partition (card A) N ∧ size N ≤ card B}*

using \langle finite A \rangle \langle finite B \rangle by (rule *bij-betw-number-partition-of*)

from *this* have $\text{card } ((A \rightarrow_E B) // \text{domain-and-range-permutation } A B) = \text{card } \{N. \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$

by (rule *bij-betw-same-card*)

also have $\text{card } \{N. \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\} = \text{Partition } (\text{card } A + \text{card } B) (\text{card } B)$

by (rule *card-number-partitions-with-atmost-k-parts*)

finally show *?thesis* .

qed

end

14 Injections from A to B up to a permutation on A and B

theory *Twelfefold-Way-Entry11*

imports *Twelfefold-Way-Entry10*

begin

14.1 Properties for Bijections

lemma *all-one-implies-inj-on*:

assumes *finite A finite B*

assumes $\forall n. n \in \# N \longrightarrow n = 1$ *number-partition (card A) N size N ≤ card B*

assumes $f \in \text{functions-of } A B N$

shows *inj-on f A*

proof –

from $\langle f \in \text{functions-of } A B N \rangle$ have $f \in A \rightarrow_E B$

and $N = \text{image-mset card (mset-set ((\lambda b. \{x \in A. f x = b\}) ' B - \{\{\}\}))}$

unfolding *functions-of-def* by *auto*

from *this* $\langle \forall n. n \in \# N \longrightarrow n = 1 \rangle$ have *parts*: $\forall b \in B. \text{card } \{x \in A. f x = b\} = 1 \vee \{x \in A. f x = b\} = \{\}$

using \langle finite B \rangle by *auto*

show *inj-on f A*

proof

fix $x y$

assume $a: x \in A y \in A f x = f y$

```

from ⟨ $f \in A \rightarrow_E B$ ⟩ ⟨ $x \in A$ ⟩ have  $f x \in B$  by auto
from  $a$  have 1:  $x \in \{x' \in A. f x' = f x\}$   $y \in \{x' \in A. f x' = f x\}$  by auto
from this have 2:  $\text{card } \{x' \in A. f x' = f x\} = 1$ 
  using parts ⟨ $f x \in B$ ⟩ by blast
from this have is-singleton  $\{x' \in A. f x' = f x\}$ 
  by (simp add: is-singleton-altdef)
from 1 this show  $x = y$ 
  by (metis is-singletonE singletonD)
qed
qed

```

lemma *inj-on-implies-all-one*:

```

assumes finite  $A$  finite  $B$ 
assumes  $F \in (A \rightarrow_E B)$  // domain-and-range-permutation  $A B$ 
assumes univ  $(\lambda f. \text{inj-on } f A) F$ 
shows  $\forall n. n \in \# \text{ number-partition-of } A B F \longrightarrow n = 1$ 
proof –
  from ⟨ $F \in (A \rightarrow_E B)$  // domain-and-range-permutation  $A B$ ⟩ obtain  $f$  where
   $f \in A \rightarrow_E B$ 
  and  $F\text{-eq}$ :  $F = \text{domain-and-range-permutation } A B \text{ “}\{f\}$  using quotientE by
  blast
  have number-partition-of  $A B F = \text{univ } (\lambda f. \text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}))) F$ 
  unfolding number-partition-of-def ..
  also have ... =  $\text{univ } (\lambda f. \text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}))) (\text{domain-and-range-permutation } A B \text{ “}\{f\})$ 
  unfolding  $F\text{-eq}$  ..
  also have ... =  $\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}))$ 
  using ⟨finite  $B$ ⟩ equiv-domain-and-range-permutation multiset-of-partition-cards-respects-domain-and-range-
  ⟨ $f \in A \rightarrow_E B$ ⟩
  by (subst univ-commute') auto
  finally have  $\text{eq}$ :  $\text{number-partition-of } A B F = \text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\}))$  .
  from iffD1[ $OF$  univ-commute',  $OF$  equiv-domain-and-range-permutation,  $OF$  inj-on-respects-domain-and-range-permutation,  $OF$  ⟨ $f \in A \rightarrow_E B$ ⟩]
  assms(4) have inj-on  $f A$  by (simp add: F-eq)
  have  $\forall n. n \in \# \text{ image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) \text{ ‘ } B - \{\{\}\})) \longrightarrow n = 1$ 
proof –
  have  $\forall b \in B. \text{card } \{x \in A. f x = b\} = 1 \vee \{x \in A. f x = b\} = \{\}$ 
proof
  fix  $b$ 
  assume  $b \in B$ 
  show  $\text{card } \{x \in A. f x = b\} = 1 \vee \{x \in A. f x = b\} = \{\}$ 
proof (cases  $b \in f \text{ ‘ } A$ )
  assume  $b \in f \text{ ‘ } A$ 
  from ⟨inj-on  $f A$ ⟩ this have is-singleton  $\{x \in A. f x = b\}$ 
  by (auto simp add: inj-on-eq-iff intro: is-singletonI')
  from this have  $\text{card } \{x \in A. f x = b\} = 1$ 

```

```

    by (subst is-singleton-altdef[symmetric])
    from this show ?thesis ..
next
  assume  $b \notin f \text{ ` } A$ 
  from this have  $\{x \in A. f x = b\} = \{\}$  by auto
  from this show ?thesis ..
qed
qed
from this show ?thesis
  using ⟨finite B⟩ by auto
qed
from this show  $\forall n. n \in \# \text{ number-partition-of } A B F \longrightarrow n = 1$ 
  unfolding eq by auto
qed

```

lemma *functions-of-is-inj-on:*

```

  assumes finite A finite B
  assumes  $\forall n. n \in \# N \longrightarrow n = 1 \text{ number-partition } (\text{card } A) N \text{ size } N \leq \text{card } B$ 
  shows univ  $(\lambda f. \text{inj-on } f A)$  (functions-of A B N)
proof –
  have functions-of A B N  $\in (A \rightarrow_E B)$  // domain-and-range-permutation A B
  using assms functions-of by auto
  from this obtain f where eq-f: functions-of A B N = domain-and-range-permutation
  A B “ {f} and  $f \in A \rightarrow_E B$ 
  using quotientE by blast
  from eq-f have  $f \in \text{functions-of } A B N$ 
  using ⟨ $f \in A \rightarrow_E B$ ⟩ equiv-domain-and-range-permutation equiv-class-self by
  fastforce
  have inj-on f A
  using ⟨ $f \in \text{functions-of } A B N$ ⟩ assms all-one-implies-inj-on by blast
  from this show ?thesis
  unfolding eq-f using equiv-domain-and-range-permutation inj-on-respects-domain-and-range-permutation
  ⟨ $f \in A \rightarrow_E B$ ⟩
  by (subst univ-commute') assumption+
qed

```

14.2 Bijections

lemma *bij-betw-number-partition-of:*

```

  assumes finite A finite B
  shows bij-betw (number-partition-of A B) ( $\{f \in A \rightarrow_E B. \text{inj-on } f A\}$  // do-
  main-and-range-permutation A B)  $\{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition}$ 
  (card A) N  $\wedge \text{size } N \leq \text{card } B\}$ 
proof (rule bij-betw-byWitness[where f'=functions-of A B])
  have quotient-eq:  $\{f \in A \rightarrow_E B. \text{inj-on } f A\}$  // domain-and-range-permutation
  A B =  $\{F \in ((A \rightarrow_E B) // \text{domain-and-range-permutation } A B). \text{univ } (\lambda f. \text{inj-on}$ 
  f A) F\}
  using equiv-domain-and-range-permutation[of A B] inj-on-respects-domain-and-range-permutation[of
  A B] by (simp only: univ-preserves-predicate)

```

show $\forall F \in \{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$.
functions-of $A B$ (*number-partition-of* $A B F$) = F
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*auto simp only: quotient-eq functions-of-number-partition-of*)
show $\forall N \in \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$. *number-partition-of* $A B$ (*functions-of* $A B N$) = N
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ *number-partition-of-functions-of* **by** *auto*
show *number-partition-of* $A B$ ' ($\{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$)
 $\subseteq \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$
by (*auto simp add: quotient-eq number-partition-of inj-on-implies-all-one simp del: One-nat-def*)
show *functions-of* $A B$ ' $\{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$
 $\subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*auto simp add: quotient-eq intro: functions-of functions-of-is-inj-on*)
qed

lemma *bij-betw-functions-of:*

assumes *finite* A *finite* B
shows *bij-betw* (*functions-of* $A B$) $\{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$ ($\{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$)
proof (*rule* *bij-betw-byWitness*[**where** $f' = \text{number-partition-of } A B$])
have *quotient-eq:* $\{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B = \{F \in ((A \rightarrow_E B) // \text{domain-and-range-permutation } A B). \text{univ } (\lambda f. \text{inj-on } f A) F\}$
using *equiv-domain-and-range-permutation*[*of* $A B$] *inj-on-respects-domain-and-range-permutation*[*of* $A B$] **by** (*simp only: univ-preserves-predicate*)
show $\forall F \in \{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$.
functions-of $A B$ (*number-partition-of* $A B F$) = F
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*auto simp only: quotient-eq functions-of-number-partition-of*)
show $\forall N \in \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$. *number-partition-of* $A B$ (*functions-of* $A B N$) = N
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ *number-partition-of-functions-of* **by** *auto*
show *number-partition-of* $A B$ ' ($\{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$)
 $\subseteq \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$
by (*auto simp add: quotient-eq number-partition-of inj-on-implies-all-one simp del: One-nat-def*)
show *functions-of* $A B$ ' $\{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$
 $\subseteq \{f \in A \rightarrow_E B. \text{inj-on } f A\}$ // domain-and-range-permutation $A B$
using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ **by** (*auto simp add: quotient-eq intro: functions-of functions-of-is-inj-on*)

qed

14.3 Cardinality

lemma *card-injective-functions-domain-and-range-permutation:*

assumes *finite A finite B*

shows $\text{card } (\{f \in A \rightarrow_E B. \text{inj-on } f A\} // \text{domain-and-range-permutation } A B)$
 $= \text{iverson } (\text{card } A \leq \text{card } B)$

proof –

have *bij-betw (number-partition-of A B) ($\{f \in A \rightarrow_E B. \text{inj-on } f A\} // \text{domain-and-range-permutation } A B)$ $\{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$*

using *$\langle \text{finite } A \rangle \langle \text{finite } B \rangle$ by (rule *bij-betw-number-partition-of*)*

from *this* **have** $\text{card } (\{f \in A \rightarrow_E B. \text{inj-on } f A\} // \text{domain-and-range-permutation } A B) = \text{card } \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\}$

by (rule *bij-betw-same-card*)

also **have** $\text{card } \{N. (\forall n. n \in \# N \longrightarrow n = 1) \wedge \text{number-partition } (\text{card } A) N \wedge \text{size } N \leq \text{card } B\} = \text{iverson } (\text{card } A \leq \text{card } B)$

by (rule *card-number-partitions-with-only-parts-1*)

finally **show** *?thesis* .

qed

end

15 Surjections from A to B up to a Permutation on A and B

theory *Twelfefold-Way-Entry12*

imports *Twelfefold-Way-Entry9 Twelfefold-Way-Entry10*

begin

15.1 Properties for Bijections

lemma *size-eq-card-implies-surj-on:*

assumes *finite A finite B*

assumes *size N = card B*

assumes *f ∈ functions-of A B N*

shows *f ‘ A = B*

proof –

from *$\langle f \in \text{functions-of } A B N \rangle$* **have** $f \in A \rightarrow_E B$ **and**

$N = \text{image-mset } \text{card } (\text{mset-set } ((\lambda b. \{x \in A. f x = b\}) ‘ B - \{\{\}\}))$

unfolding *functions-of-def* **by** *auto*

from *this* **size N = card B** **have** $\text{card } ((\lambda b. \{x \in A. f x = b\}) ‘ B - \{\{\}\}) = \text{card } B$ **by** *simp*

from *this* *$\langle \text{finite } B \rangle$* **show** $f ‘ A = B$

using *card-eq-implies-surjective-on* **by** *blast*

qed

lemma *surj-on-implies-size-eq-card*:

assumes *finite A finite B*

assumes $F \in (A \rightarrow_E B)$ // *domain-and-range-permutation A B*

assumes *univ* $(\lambda f. f \text{ ‘ } A = B)$ *F*

shows *size* $(\text{number-partition-of } A \ B \ F) = \text{card } B$

proof –

from $\langle F \in (A \rightarrow_E B) \text{ // domain-and-range-permutation } A \ B \rangle$ **obtain** *f* **where**
 $f \in A \rightarrow_E B$

and *F-eq*: $F = \text{domain-and-range-permutation } A \ B \text{ ‘ ‘ } \{f\}$ **using** *quotientE* **by**
blast

have *number-partition-of A B F* = *univ* $(\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\})$ *F*

unfolding *number-partition-of-def* ..

also have ... = *univ* $(\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\})$ *(domain-and-range-permutation A B ‘ ‘ {f})*

unfolding *F-eq* ..

also have ... = *image-mset card* $(\text{mset-set } ((\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\}))$

using $\langle \text{finite } B \rangle$ *equiv-domain-and-range-permutation multiset-of-partition-cards-respects-domain-and-range-*
 $\langle f \in A \rightarrow_E B \rangle$

by *(subst univ-commute’)* *auto*

finally have *eq*: *number-partition-of A B F* = *image-mset card* $(\text{mset-set } ((\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\}))$.

from *iffD1* [*OF univ-commute’*, *OF equiv-domain-and-range-permutation*, *OF*
surjective-respects-domain-and-range-permutation, *OF* $\langle f \in A \rightarrow_E B \rangle$]

assms(4) **have** $f \text{ ‘ } A = B$ **by** *(simp add: F-eq)*

have *size* $(\text{number-partition-of } A \ B \ F) = \text{size } (\text{image-mset card } (\text{mset-set } ((\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\})))$

unfolding *eq* ..

also have ... = *card* $((\lambda b. \{x \in A. f \ x = b\}) \text{ ‘ } B - \{\{\}\})$ **by** *simp*

also from $\langle f \text{ ‘ } A = B \rangle$ **have** ... = *card B*

using *surjective-on-implies-card-eq* **by** *auto*

finally show *?thesis* .

qed

lemma *functions-of-is-surj-on*:

assumes *finite A finite B*

assumes *number-partition* $(\text{card } A) \ N \ \text{size } N = \text{card } B$

shows *univ* $(\lambda f. f \text{ ‘ } A = B)$ *(functions-of A B N)*

proof –

have *functions-of A B N* $\in (A \rightarrow_E B)$ // *domain-and-range-permutation A B*

using *functions-of* $\langle \text{finite } A \rangle \langle \text{finite } B \rangle \langle \text{number-partition } (\text{card } A) \ N \ \text{size } N = \text{card } B \rangle$

by *fastforce*

from this obtain *f* **where** *eq-f*: *functions-of A B N* = *domain-and-range-permutation*
 $A \ B \text{ ‘ ‘ } \{f\}$ **and** $f \in A \rightarrow_E B$

using *quotientE* **by** *blast*

from *eq-f* **have** $f \in \text{functions-of } A \ B \ N$

using $\langle f \in A \rightarrow_E B \rangle$ *equiv-domain-and-range-permutation equiv-class-self* **by**
fastforce

```

have f ' A = B
  using ⟨f ∈ functions-of A B N⟩ assms size-eq-card-implies-surj-on by blast
from this show ?thesis
  unfolding eq-f using equiv-domain-and-range-permutation surjective-respects-domain-and-range-permutation
⟨f ∈ A →E B⟩
  by (subst univ-commute') assumption+
qed

```

15.2 Bijections

lemma *bij-betw-number-partition-of*:

```

  assumes finite A finite B
  shows bij-betw (number-partition-of A B) ({f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B) {N. number-partition (card A) N ∧ size N = card B}

```

proof (*rule bij-betw-byWitness*[**where** *f'=functions-of A B*])

```

  have quotient-eq: {f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B = {F ∈ ((A →E B) // domain-and-range-permutation A B). univ (λf. f ' A = B) F}

```

```

  using equiv-domain-and-range-permutation[of A B] surjective-respects-domain-and-range-permutation[of A B] by (simp only: univ-preserves-predicate)

```

```

  show ∀F∈{f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B.

```

```

    functions-of A B (number-partition-of A B F) = F

```

```

  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp only: quotient-eq functions-of-number-partition-of)

```

```

  show ∀N∈{N. number-partition (card A) N ∧ size N = card B}. number-partition-of A B (functions-of A B N) = N

```

```

  using ⟨finite A⟩ ⟨finite B⟩ by (simp add: number-partition-of-functions-of)

```

```

  show number-partition-of A B ' ({f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B)

```

```

    ⊆ {N. number-partition (card A) N ∧ size N = card B}

```

```

  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp add: quotient-eq number-partition-of surj-on-implies-size-eq-card)

```

```

  show functions-of A B ' {N. number-partition (card A) N ∧ size N = card B}

```

```

    ⊆ {f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B

```

```

  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp add: quotient-eq intro: functions-of functions-of-is-surj-on)

```

qed

lemma *bij-betw-functions-of*:

```

  assumes finite A finite B

```

```

  shows bij-betw (functions-of A B) {N. number-partition (card A) N ∧ size N = card B} ({f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B)

```

proof (*rule bij-betw-byWitness*[**where** *f'=number-partition-of A B*])

```

  have quotient-eq: {f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B = {F ∈ ((A →E B) // domain-and-range-permutation A B). univ (λf. f ' A = B) F}

```

```

  using equiv-domain-and-range-permutation[of A B] surjective-respects-domain-and-range-permutation[of A B] by (simp only: univ-preserves-predicate)

```

```

  show ∀F∈{f ∈ A →E B. f ' A = B} // domain-and-range-permutation A B.

```

```

    functions-of A B (number-partition-of A B F) = F
  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp only: quotient-eq functions-of-number-partition-of)
  show  $\forall N \in \{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N = \text{card } B\}. \text{number-partition-of}$ 
  A B (functions-of A B N) = N
  using ⟨finite A⟩ ⟨finite B⟩ by (simp add: number-partition-of-functions-of)
  show number-partition-of A B ‘ $\{f \in A \rightarrow_E B. f ‘ A = B\}$  // domain-and-range-permutation
  A B)
   $\subseteq \{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N = \text{card } B\}$ 
  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp add: quotient-eq number-partition-of
  surj-on-implies-size-eq-card)
  show functions-of A B ‘ $\{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N = \text{card } B\}$ 
   $\subseteq \{f \in A \rightarrow_E B. f ‘ A = B\}$  // domain-and-range-permutation A B
  using ⟨finite A⟩ ⟨finite B⟩ by (auto simp add: quotient-eq intro: functions-of
  functions-of-is-surj-on)
qed

```

15.3 Cardinality

lemma *card-surjective-functions-domain-and-range-permutation:*

```

  assumes finite A finite B
  shows card  $\{f \in A \rightarrow_E B. f ‘ A = B\}$  // domain-and-range-permutation A B)
  = Partition (card A) (card B)
proof –
  have bij-betw (number-partition-of A B)  $\{f \in A \rightarrow_E B. f ‘ A = B\}$  // do-
  main-and-range-permutation A B)  $\{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N =$ 
  card B}
  using ⟨finite A⟩ ⟨finite B⟩ by (rule bij-betw-number-partition-of)
  from this have card  $\{f \in A \rightarrow_E B. f ‘ A = B\}$  // domain-and-range-permutation
  A B) = card  $\{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N = \text{card } B\}$ 
  by (rule bij-betw-same-card)
  also have card  $\{N. \text{number-partition} (\text{card } A) N \wedge \text{size } N = \text{card } B\} = \text{Partition}$ 
  (card A) (card B)
  by (rule card-partitions-with-k-parts)
  finally show ?thesis .
qed

```

end

16 Cardinality of Bijections

theory *Card-Bijections*

imports

```

  Twelfold-Way-Entry2
  Twelfold-Way-Entry3
  Twelfold-Way-Entry5
  Twelfold-Way-Entry6
  Twelfold-Way-Entry8
  Twelfold-Way-Entry9
  Twelfold-Way-Entry11

```


Twelfold-Way-Entry12

begin

16.1 Bijections from A to B

lemma *bij-betw-set-is-empty*:

assumes *finite A finite B*

assumes *card A \neq card B*

shows $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \{\}$

using *assms bij-betw-same-card* **by** *blast*

lemma *card-bijections-eq-zero*:

assumes *finite A finite B*

assumes *card A \neq card B*

shows *card* $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = 0$

using *bij-betw-set-is-empty[OF assms]* **by** (*simp only: card.empty*)

Two alternative proofs for the cardinality of bijections up to a permutation on A.

lemma

assumes *finite A finite B*

assumes *card A = card B*

shows *card* $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \text{fact } (\text{card } B)$

proof –

have *card* $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \text{card } \{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$

using $\langle \text{finite } B \rangle \langle \text{card } A = \text{card } B \rangle$ **by** (*metis bij-betw-implies-inj-on-and-card-eq*)

also have $\dots = \text{fact } (\text{card } B)$

using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle \langle \text{card } A = \text{card } B \rangle$ **by** (*simp add: card-extensional-funcset-inj-on*)

finally show *?thesis* .

qed

lemma *card-bijections*:

assumes *finite A finite B*

assumes *card A = card B*

shows *card* $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \text{fact } (\text{card } B)$

proof –

have *card* $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \text{card } \{f \in A \rightarrow_E B. f \ ' \ A = B\}$

using $\langle \text{finite } A \rangle \langle \text{card } A = \text{card } B \rangle$

by (*metis bij-betw-implies-surj-on-and-card-eq*)

also have $\dots = \text{fact } (\text{card } B)$

using $\langle \text{finite } A \rangle \langle \text{finite } B \rangle \langle \text{card } A = \text{card } B \rangle$

by (*simp add: card-extensional-funcset-surj-on*)

finally show *?thesis* .

qed

16.2 Bijections from A to B up to a Permutation on A

lemma *bij-betw-quotient-domain-permutation-eq-empty*:

assumes *card A \neq card B*

shows $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B = \{\}$
using $\langle \text{card } A \neq \text{card } B \rangle \text{bij-betw-same-card}$ **by** *auto*

lemma *card-bijections-domain-permutation-eq-0:*

assumes $\text{card } A \neq \text{card } B$
shows $\text{card } (\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B) = 0$
using *bij-betw-quotient-domain-permutation-eq-empty*[*OF assms*] **by** (*simp only: card.empty*)

Two alternative proofs for the cardinality of bijections up to a permutation on A.

lemma

assumes *finite A finite B*
assumes $\text{card } A = \text{card } B$
shows $\text{card } (\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B) = 1$
proof –
from *assms* **have** $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B$
 $= \{f \in A \rightarrow_E B. \text{inj-on } f \ A\} // \text{domain-permutation } A \ B$
by (*metis (no-types, lifting) PiE-cong bij-betw-implies-inj-on-and-card-eq*)
from this show *?thesis*
using *assms* **by** (*simp add: card-injective-functions-domain-permutation*)
qed

lemma *card-bijections-domain-permutation-eq-1:*

assumes *finite A finite B*
assumes $\text{card } A = \text{card } B$
shows $\text{card } (\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B) = 1$
proof –
from *assms* **have** $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B$
 $= \{f \in A \rightarrow_E B. f \ ' \ A = B\} // \text{domain-permutation } A \ B$
by (*metis (no-types, lifting) PiE-cong bij-betw-implies-surj-on-and-card-eq*)
from this show *?thesis*
using *assms* **by** (*simp add: card-surjective-functions-domain-permutation*)
qed

lemma *card-bijections-domain-permutation:*

assumes *finite A finite B*
shows $\text{card } (\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{domain-permutation } A \ B) =$
iverson (card A = card B)
using *assms card-bijections-domain-permutation-eq-0 card-bijections-domain-permutation-eq-1*
unfolding *iverson-def* **by** *auto*

16.3 Bijections from A to B up to a Permutation on B

lemma *bij-betw-quotient-range-permutation-eq-empty:*

assumes $\text{card } A \neq \text{card } B$
shows $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} // \text{range-permutation } A \ B = \{\}$
using $\langle \text{card } A \neq \text{card } B \rangle \text{bij-betw-same-card}$ **by** *auto*

lemma *card-bijections-range-permutation-eq-0*:
assumes *card A ≠ card B*
shows *card ({f ∈ A →_E B. bij-betw f A B} // range-permutation A B) = 0*
using *bij-betw-quotient-range-permutation-eq-empty*[*OF assms*] **by** (*simp only: card.empty*)

Two alternative proofs for the cardinality of bijections up to a permutation on B.

lemma
assumes *finite A finite B*
assumes *card A = card B*
shows *card ({f ∈ A →_E B. bij-betw f A B} // range-permutation A B) = 1*
proof –
from *assms* **have** *{f ∈ A →_E B. bij-betw f A B} // range-permutation A B = {f ∈ A →_E B. inj-on f A} // range-permutation A B*
by (*metis (no-types, lifting) PiE-cong bij-betw-implies-inj-on-and-card-eq*)
from *this* **show** *?thesis*
using *assms* **by** (*simp add: iverson-def card-injective-functions-range-permutation*)
qed

lemma *card-bijections-range-permutation-eq-1*:
assumes *finite A finite B*
assumes *card A = card B*
shows *card ({f ∈ A →_E B. bij-betw f A B} // range-permutation A B) = 1*
proof –
from *assms* **have** *{f ∈ A →_E B. bij-betw f A B} // range-permutation A B = {f ∈ A →_E B. f ‘ A = B} // range-permutation A B*
by (*metis (no-types, lifting) PiE-cong bij-betw-implies-surj-on-and-card-eq*)
from *this* **show** *?thesis*
using *assms* **by** (*simp add: card-surjective-functions-range-permutation*)
qed

lemma *card-bijections-range-permutation*:
assumes *finite A finite B*
shows *card ({f ∈ A →_E B. bij-betw f A B} // range-permutation A B) = iverson (card A = card B)*
using *assms card-bijections-range-permutation-eq-0 card-bijections-range-permutation-eq-1*
unfolding *iverson-def* **by** *auto*

16.4 Bijections from A to B up to a Permutation on A and B

lemma *bij-betw-quotient-domain-and-range-permutation-eq-empty*:
assumes *card A ≠ card B*
shows *{f ∈ A →_E B. bij-betw f A B} // domain-and-range-permutation A B = {}*
using *⟨card A ≠ card B⟩ bij-betw-same-card* **by** *auto*

lemma *card-bijections-domain-and-range-permutation-eq-0*:
assumes *card A ≠ card B*

```

shows card ( $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B) = 0
using bij-betw-quotient-domain-and-range-permutation-eq-empty[OF assms] by (simp
only: card.empty)

```

Two alternative proofs for the cardinality of bijections up to a permutation on A and B.

lemma

```

assumes finite A finite B
assumes card A = card B
shows card ( $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B) = 1
proof -
from assms have  $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B =
   $\{f \in A \rightarrow_E B. \text{inj-on } f \ A\}$  // domain-and-range-permutation A B
by (metis (no-types, lifting) PiE-cong bij-betw-implies-inj-on-and-card-eq)
from this show ?thesis
using assms by (simp add: iverson-def card-injective-functions-domain-and-range-permutation)
qed

```

lemma card-bijections-domain-and-range-permutation-eq-1:

```

assumes finite A finite B
assumes card A = card B
shows card ( $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B) = 1
proof -
from assms have  $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B =
   $\{f \in A \rightarrow_E B. f \ A = B\}$  // domain-and-range-permutation A B
by (metis (no-types, lifting) PiE-cong bij-betw-implies-surj-on-and-card-eq)
from this show ?thesis
using assms by (simp add: card-surjective-functions-domain-and-range-permutation
Partition-diag)
qed

```

lemma card-bijections-domain-and-range-permutation:

```

assumes finite A finite B
shows card ( $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B) = iverson (card A = card B)
using assms card-bijections-domain-and-range-permutation-eq-0 card-bijections-domain-and-range-permutation
unfolding iverson-def by auto

```

end

17 Direct Proofs for Cardinality of Bijections

theory Card-Bijections-Direct

imports

begin

17.1 Bijections from A to B up to a Permutation on A

17.1.1 Equivalence Class

lemma *bijections-in-domain-permutation:*

assumes *finite A finite B*

assumes *card A = card B*

shows $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} \in \{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} //$
domain-permutation A B

proof –

from *assms* **obtain** *f* **where** $f: f \in \{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$

by (*metis finite-same-card-bij-on-ext-funcset mem-Collect-eq*)

moreover **have** *proj-f*: $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} = \text{domain-permutation } A \ B \text{ “ } \{f\}$

proof

from *f* **show** $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\} \subseteq \text{domain-permutation } A \ B \text{ “ } \{f\}$

unfolding *domain-permutation-def*

by (*auto elim: obtain-domain-permutation-for-two-bijections*)

next

show *domain-permutation A B “ {f} ⊆ {f ∈ A →_E B. bij-betw f A B}*

proof

fix *f'*

assume $f' \in \text{domain-permutation } A \ B \text{ “ } \{f\}$

have $(f', f) \in \text{domain-permutation } A \ B$

using $\langle f' \in \text{domain-permutation } A \ B \text{ “ } \{f\} \rangle$ *equiv-domain-permutation[of*
A B]

by (*simp add: equiv-class-eq-iff*)

from *this* **obtain** *p* **where** *p* *permutes A* $\forall x \in A. f' \ x = f \ (p \ x)$

unfolding *domain-permutation-def* **by** *auto*

from *this* **have** *bij-betw (f ∘ p) A B*

using *bij-betw-comp-iff f permutes-imp-bij* **by** *fastforce*

from *this* **have** *bij-betw f' A B*

using $\langle \forall x \in A. f' \ x = f \ (p \ x) \rangle$

by (*metis (mono-tags, lifting) bij-betw-cong comp-apply*)

moreover **have** $f' \in A \rightarrow_E B$

using $\langle f' \in \text{domain-permutation } A \ B \text{ “ } \{f\} \rangle$

unfolding *domain-permutation-def* **by** *auto*

ultimately **show** $f' \in \{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$ **by** *simp*

qed

qed

ultimately **show** *?thesis* **by** (*simp add: quotientI*)

qed

lemma *bij-betw-quotient-domain-permutation-eq:*

assumes *finite A finite B*

assumes *card A = card B*

```

shows { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } // domain-permutation  $A$   $B$  = {{ $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }}
proof
  show {{ $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }}  $\subseteq$  { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } // domain-permutation  $A$   $B$ 
    by (simp add: bijections-in-domain-permutation[OF assms])
next
  show { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } // domain-permutation  $A$   $B$   $\subseteq$  {{ $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }}
proof
  fix  $F$ 
  assume  $F$ -in:  $F \in$  { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } // domain-permutation  $A$   $B$ 
  have { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } // domain-permutation  $A$   $B$  = { $F \in ((A \rightarrow_E B) // \text{domain-permutation } A B)$ . univ ( $\lambda f$ . bij-betw  $f$   $A$   $B$ )  $F$ }
  using equiv-domain-permutation[of A B] bij-betw-respects-domain-permutation[of A B] by (simp only: univ-preserves-predicate)
  from  $F$ -in this have  $F \in (A \rightarrow_E B)$  // domain-permutation  $A$   $B$ 
  and univ ( $\lambda f$ . bij-betw  $f$   $A$   $B$ )  $F$ 
  by blast+
have  $F =$  { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }
proof
  have  $\forall f \in F$ .  $f \in A \rightarrow_E B$ 
  using  $\langle F \in (A \rightarrow_E B) // \text{domain-permutation } A B \rangle$ 
  by (metis ImageE equiv-class-eq-iff equiv-domain-permutation quotientE)
  moreover have  $\forall f \in F$ . bij-betw  $f$   $A$   $B$ 
  using univ-predicate-impl-forall[OF equiv-domain-permutation bij-betw-respects-domain-permutation]
  using  $\langle F \in (A \rightarrow_E B) // \text{domain-permutation } A B \rangle$   $\langle \text{univ } (\lambda f$ . bij-betw  $f$   $A$   $B$ )  $F \rangle$ 
by auto
  ultimately show  $F \subseteq$  { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ } by auto
next
show { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }  $\subseteq$   $F$ 
proof
  fix  $f'$ 
  assume  $f' \in$  { $f \in A \rightarrow_E B$ . bij-betw  $f$   $A$   $B$ }
  from this have  $f' \in A \rightarrow_E B$  bij-betw  $f'$   $A$   $B$  by auto
  obtain  $f$  where  $f \in A \rightarrow_E B$  and  $F = \text{domain-permutation } A B$  “ { $f$ }
  using  $\langle F \in (A \rightarrow_E B) // \text{domain-permutation } A B \rangle$  by (auto elim: quotientE)
  have bij-betw  $f$   $A$   $B$ 
  using univ-commute'[OF equiv-domain-permutation bij-betw-respects-domain-permutation]
  using  $\langle f \in A \rightarrow_E B \rangle$   $\langle F = \text{domain-permutation } A B$  “ { $f$ }  $\rangle$   $\langle \text{univ } (\lambda f$ . bij-betw  $f$   $A$   $B$ )  $F \rangle$ 
by auto
  obtain  $p$  where  $p$  permutes  $A$   $\forall x \in A$ .  $f x = f' (p x)$ 
  using obtain-domain-permutation-for-two-bijections
  using  $\langle \text{bij-betw } f$   $A$   $B \rangle$   $\langle \text{bij-betw } f'$   $A$   $B \rangle$  by blast
  from this  $\langle f \in A \rightarrow_E B \rangle$   $\langle f' \in A \rightarrow_E B \rangle$ 
  have  $(f, f') \in \text{domain-permutation } A B$ 

```

```

    unfolding domain-permutation-def by auto
  from this show  $f' \in F$ 
    using  $\langle F = \text{domain-permutation } A B \text{ “ } \{f\} \rangle$  by simp
qed
qed
from this show  $F \in \{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\}\}$  by simp
qed
qed

```

17.1.2 Cardinality

```

lemma
  assumes finite A finite B
  assumes card A = card B
  shows card  $\{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{domain-permutation } A B\} = 1$ 
  using bij-betw-quotient-domain-permutation-eq[OF assms] by auto

```

17.2 Bijections from A to B up to a Permutation on B

17.2.1 Equivalence Class

```

lemma bijections-in-range-permutation:
  assumes finite A finite B
  assumes card A = card B
  shows  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} \in \{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B\}$ 
proof -
  from assms obtain f where  $f: f \in \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$ 
  by (metis finite-same-card-bij-on-ext-funcset mem-Collect-eq)
  moreover have  $\text{proj-}f: \{f \in A \rightarrow_E B. \text{bij-betw } f A B\} = \text{range-permutation } A B \text{ “ } \{f\}$ 
  proof
    from f show  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} \subseteq \text{range-permutation } A B \text{ “ } \{f\}$ 
    unfolding range-permutation-def
    by (auto elim: obtain-range-permutation-for-two-bijections)
  next
    show  $\text{range-permutation } A B \text{ “ } \{f\} \subseteq \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$ 
    proof
      fix f'
      assume  $f' \in \text{range-permutation } A B \text{ “ } \{f\}$ 
      have  $(f', f) \in \text{range-permutation } A B$ 
      using  $\langle f' \in \text{range-permutation } A B \text{ “ } \{f\} \rangle$  equiv-range-permutation[of A B]
      by (simp add: equiv-class-eq-iff)
      from this obtain p where  $p$  permutes  $B \ \forall x \in A. f' x = p (f x)$ 
      unfolding range-permutation-def by auto
      from this have  $\text{bij-betw } (p \circ f) A B$ 
      using bij-betw-comp-iff  $f$  permutes-imp-bij by fastforce
      from this have  $\text{bij-betw } f' A B$ 
      using  $\langle \forall x \in A. f' x = p (f x) \rangle$ 
      by (metis (mono-tags, lifting) bij-betw-cong comp-apply)
    qed
  qed

```

```

moreover have  $f' \in A \rightarrow_E B$ 
  using  $\langle f' \in \text{range-permutation } A B \text{ “}\{f\}\rangle$ 
  unfolding range-permutation-def by auto
  ultimately show  $f' \in \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$  by simp
qed
qed
ultimately show ?thesis by (simp add: quotientI)
qed

```

lemma *bij-betw-quotient-range-permutation-eq*:

```

assumes finite A finite B
assumes  $\text{card } A = \text{card } B$ 

```

```

shows  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B = \{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\}\}$ 

```

proof

```

show  $\{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\}\} \subseteq \{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B$ 

```

```

  by (simp add: bijections-in-range-permutation[OF assms])

```

next

```

show  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B \subseteq \{\{f \in A \rightarrow_E B. \text{bij-betw } f A B\}\}$ 

```

proof

```

fix  $F$ 

```

```

assume F-in:  $F \in \{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B$ 

```

```

have  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} // \text{range-permutation } A B = \{F \in ((A \rightarrow_E B) // \text{range-permutation } A B). \text{univ } (\lambda f. \text{bij-betw } f A B) F\}$ 

```

```

using equiv-range-permutation[of A B] bij-betw-respects-range-permutation[of A B] by (simp only: univ-preserves-predicate)

```

```

from this F-in have  $F \in (A \rightarrow_E B) // \text{range-permutation } A B$ 

```

```

  and univ  $(\lambda f. \text{bij-betw } f A B) F$  by blast+

```

```

have  $F = \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$ 

```

proof

```

have  $\forall f \in F. f \in A \rightarrow_E B$ 

```

```

  using  $\langle F \in (A \rightarrow_E B) // \text{range-permutation } A B \rangle$ 

```

```

  by (metis ImageE equiv-class-eq-iff equiv-range-permutation quotientE)

```

```

moreover have  $\forall f \in F. \text{bij-betw } f A B$ 

```

```

using univ-predicate-impl-forall[OF equiv-range-permutation bij-betw-respects-range-permutation]

```

```

using  $\langle F \in (A \rightarrow_E B) // \text{range-permutation } A B \rangle \langle \text{univ } (\lambda f. \text{bij-betw } f A B) F \rangle$ 

```

```

  by auto

```

```

ultimately show  $F \subseteq \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$  by auto

```

next

```

show  $\{f \in A \rightarrow_E B. \text{bij-betw } f A B\} \subseteq F$ 

```

proof

```

fix  $f'$ 

```

```

assume  $f' \in \{f \in A \rightarrow_E B. \text{bij-betw } f A B\}$ 

```

```

from this have  $f' \in A \rightarrow_E B \text{bij-betw } f' A B$  by auto

```

```

  obtain  $f$  where  $f \in A \rightarrow_E B$  and  $F = \text{range-permutation } A B \text{ “}\{f\}$ 

```

```

using  $\langle F \in (A \rightarrow_E B) // \text{range-permutation } A B \rangle$  by (auto elim: quotientE)

```



```

    have bij-betw f A B
    using univ-commute'[OF equiv-range-permutation bij-betw-respects-range-permutation]
      using ⟨f ∈ A →E B⟩ ⟨F = range-permutation A B “ {f}⟩ ⟨univ (λf.
bij-betw f A B) F⟩
      by auto
    obtain p where p permutes B ∀x∈A. f x = p (f' x)
    using obtain-range-permutation-for-two-bijections
    using ⟨bij-betw f A B⟩ ⟨bij-betw f' A B⟩ by blast
    from this ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩
    have (f, f') ∈ range-permutation A B
    unfolding range-permutation-def by auto
    from this show f' ∈ F
    using ⟨F = range-permutation A B “ {f}⟩ by simp
  qed
qed
from this show F ∈ {{f ∈ A →E B. bij-betw f A B}} by simp
qed
qed

```

17.2.2 Cardinality

lemma *card-bijections-range-permutation-eq-1:*

```

  assumes finite A finite B
  assumes card A = card B
  shows card ({f ∈ A →E B. bij-betw f A B} // range-permutation A B) = 1
  using bij-betw-quotient-range-permutation-eq[OF assms] by auto

```

17.3 Bijections from A to B up to a Permutation on A and B

17.3.1 Equivalence Class

lemma *bijections-in-domain-and-range-permutation:*

```

  assumes finite A finite B
  assumes card A = card B
  shows {f ∈ A →E B. bij-betw f A B} ∈ {f ∈ A →E B. bij-betw f A B} //
domain-and-range-permutation A B
proof –
  from assms obtain f where f: f ∈ {f ∈ A →E B. bij-betw f A B}
  by (metis finite-same-card-bij-on-ext-funcset mem-Collect-eq)
  moreover have proj-f: {f ∈ A →E B. bij-betw f A B} = domain-and-range-permutation
A B “ {f}
  proof
    have id permutes A by (simp add: permutes-id)
    from f this show {f ∈ A →E B. bij-betw f A B} ⊆ domain-and-range-permutation
A B “ {f}
    unfolding domain-and-range-permutation-def
    by (fastforce elim: obtain-range-permutation-for-two-bijections)
  next

```

```

show domain-and-range-permutation A B “ {f} ⊆ {f ∈ A →E B. bij-betw f A
B}
proof
  fix f'
  assume f' ∈ domain-and-range-permutation A B “ {f}
  have (f', f) ∈ domain-and-range-permutation A B
  using ⟨f' ∈ domain-and-range-permutation A B “ {f}⟩ equiv-domain-and-range-permutation[of
A B]
  by (simp add: equiv-class-eq-iff)
  from this obtain pA pB where pA permutes A pB permutes B
  and ∀ x ∈ A. f' x = pB (f (pA x))
  unfolding domain-and-range-permutation-def by auto
  from this have bij-betw (pB ∘ f ∘ pA) A B
  using bij-betw-comp-iff f permutes-imp-bij
  by (metis (no-types, lifting) mem-Collect-eq)
  from this have bij-betw f' A B
  using ⟨∀ x ∈ A. f' x = pB (f (pA x))⟩
  by (auto intro: bij-betw-congI)
  moreover have f' ∈ A →E B
  using ⟨f' ∈ domain-and-range-permutation A B “ {f}⟩
  unfolding domain-and-range-permutation-def by auto
  ultimately show f' ∈ {f ∈ A →E B. bij-betw f A B} by simp
qed
qed
ultimately show ?thesis by (simp add: quotientI)
qed

lemma bij-betw-quotient-domain-and-range-permutation-eq:
  assumes finite A finite B
  assumes card A = card B
  shows {f ∈ A →E B. bij-betw f A B} // domain-and-range-permutation A B =
  {{f ∈ A →E B. bij-betw f A B}}
proof
  show {{f ∈ A →E B. bij-betw f A B}}
  ⊆ {f ∈ A →E B. bij-betw f A B} // domain-and-range-permutation A B
  using bijections-in-domain-and-range-permutation[OF assms] by auto
next
  show {f ∈ A →E B. bij-betw f A B} // domain-and-range-permutation A B ⊆
  {{f ∈ A →E B. bij-betw f A B}}
  proof
    fix F
    assume F-in: F ∈ {f ∈ A →E B. bij-betw f A B} // domain-and-range-permutation
A B
    have {f ∈ A →E B. bij-betw f A B} // domain-and-range-permutation A B =
  {F ∈ ((A →E B) // domain-and-range-permutation A B). univ (λf. bij-betw f A
B) F}
    using equiv-domain-and-range-permutation[of A B] bij-betw-respects-domain-and-range-permutation[of
A B] by (simp only: univ-preserves-predicate)
    from F-in this have F ∈ (A →E B) // domain-and-range-permutation A B

```

```

    and univ (λf. bij-betw f A B) F by blast+
  have F = {f ∈ A →E B. bij-betw f A B}
  proof
    have ∀f ∈ F. f ∈ A →E B
      using ⟨F ∈ (A →E B) // domain-and-range-permutation A B⟩
      by (metis ImageE equiv-class-eq-iff equiv-domain-and-range-permutation
quotientE)
    moreover have ∀f ∈ F. bij-betw f A B
      using univ-predicate-impl-forall[OF equiv-domain-and-range-permutation
bij-betw-respects-domain-and-range-permutation]
      using ⟨F ∈ (A →E B) // domain-and-range-permutation A B⟩ ⟨univ (λf.
bij-betw f A B) F⟩
      by auto
    ultimately show F ⊆ {f ∈ A →E B. bij-betw f A B} by auto
  next
  show {f ∈ A →E B. bij-betw f A B} ⊆ F
  proof
    fix f'
    assume f' ∈ {f ∈ A →E B. bij-betw f A B}
    from this have f' ∈ A →E B bij-betw f' A B by auto
    obtain f where f ∈ A →E B and F = domain-and-range-permutation A
B “ {f}
      using ⟨F ∈ (A →E B) // domain-and-range-permutation A B⟩ by (auto
elim: quotientE)
    have bij-betw f A B
      using univ-commute'[OF equiv-domain-and-range-permutation bij-betw-respects-domain-and-range-perm
      using ⟨f ∈ A →E B⟩ ⟨F = domain-and-range-permutation A B “ {f}⟩
    ⟨univ (λf. bij-betw f A B) F⟩
      by auto
    obtain p where p permutes A ∀x∈A. f x = f' (p x)
      using obtain-domain-permutation-for-two-bijections
      using ⟨bij-betw f A B⟩ ⟨bij-betw f' A B⟩ by blast
    moreover have id permutes B by (simp add: permutes-id)
    moreover note ⟨f ∈ A →E B⟩ ⟨f' ∈ A →E B⟩
    ultimately have (f, f') ∈ domain-and-range-permutation A B
      unfolding domain-and-range-permutation-def id-def by auto
    from this show f' ∈ F
      using ⟨F = domain-and-range-permutation A B “ {f}⟩ by simp
  qed
  qed
  from this show F ∈ {{f ∈ A →E B. bij-betw f A B}} by simp
  qed
  qed

```

17.3.2 Cardinality

lemma *card-bijections-domain-and-range-permutation-eq-1:*

assumes *finite A finite B*
 assumes *card A = card B*

```

shows card ( $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$  // domain-and-range-permutation
A B) = 1
using bij-betw-quotient-domain-and-range-permutation-eq[OF assms] by auto

end

```

18 The Twelfold Way

```

theory Twelfold-Way
imports
  Preliminaries
  Twelfold-Way-Core
  Equiv-Relations-on-Functions
  Twelfold-Way-Entry1
  Twelfold-Way-Entry2
  Twelfold-Way-Entry4
  Twelfold-Way-Entry5
  Twelfold-Way-Entry6
  Twelfold-Way-Entry7
  Twelfold-Way-Entry8
  Twelfold-Way-Entry9
  Twelfold-Way-Entry3
  Twelfold-Way-Entry10
  Twelfold-Way-Entry11
  Twelfold-Way-Entry12
  Card-Bijections
  Card-Bijections-Direct
begin

end

```

References

- [1] K. P. Bogart. *Combinatorics Through Guided Discovery*. 2004.
- [2] L. Bulwahn. Cardinality of set partitions. *Archive of Formal Proofs*, Dec. 2015. http://isa-afp.org/entries/Card_Partitions.shtml, Formal proof development.
- [3] L. Bulwahn. Cardinality of multisets. *Archive of Formal Proofs*, June 2016. http://isa-afp.org/entries/Card_Multisets.shtml, Formal proof development.
- [4] L. Bulwahn. Cardinality of number partitions. *Archive of Formal Proofs*, Jan. 2016. http://isa-afp.org/entries/Card_Number_Partitions.shtml, Formal proof development.

- [5] R. P. Stanley. *Enumerative Combinatorics. Volume 1*. Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, New York, second edition, 2012.
- [6] Wikipedia. Twelfefold way — wikipedia, the free encyclopedia, 2016. [Online; accessed 4-October-2016].