

# Design Theory

Chelsea Edmonds and Lawrence Paulson

December 14, 2021

## Abstract

Combinatorial design theory studies incidence set systems with certain balance and symmetry properties. It is closely related to hypergraph theory. This formalisation presents a general library for formal reasoning on incidence set systems, designs and their applications, including formal definitions and proofs for many key properties, operations, and theorems on the construction and existence of designs. Notably, this includes formalising  $t$ -designs, balanced incomplete block designs (BIBD), group divisible designs (GDD), pairwise balanced designs (PBD), design isomorphisms, and the relationship between graphs and designs. A locale-centric approach has been used to manage the relationships between the many different types of designs. Theorems of particular interest include the necessary conditions for existence of a BIBD, Wilson's construction on GDDs, and Bose's inequality on resolvable designs. This formalisation is partly presented in the paper "A Modular First Formalisation of Combinatorial Design Theory", presented at CISM 2021.

## Contents

<b>1</b>	<b>Micellaneous Helper Functions on Sets and Multisets</b>	<b>3</b>
1.1	Set Theory Extras . . . . .	3
1.2	Multiset Helpers . . . . .	5
1.3	Partitions on Multisets . . . . .	10
<b>2</b>	<b>Design Theory Basics</b>	<b>14</b>
2.1	Initial setup . . . . .	14
2.2	Incidence System . . . . .	14
2.3	Finite Incidence Systems . . . . .	15
2.4	Designs . . . . .	16
2.5	Core Property Definitions . . . . .	17
2.5.1	Replication Number . . . . .	17
2.5.2	Point Index . . . . .	18
2.5.3	Intersection Number . . . . .	20
2.6	Incidence System Set Property Definitions . . . . .	21

2.7	Basic Constructions on designs . . . . .	23
2.7.1	Design Complements . . . . .	23
2.7.2	Multiples . . . . .	25
2.8	Simple Designs . . . . .	26
2.9	Proper Designs . . . . .	27
<b>3</b>	<b>Design Operations</b>	<b>28</b>
3.1	Incidence system definitions . . . . .	28
3.2	Incidence System Interpretations . . . . .	32
3.3	Operation Closure for Designs . . . . .	33
3.4	Combining Set Systems . . . . .	35
<b>4</b>	<b>Block and Balanced Designs</b>	<b>37</b>
4.1	Block Designs . . . . .	37
4.1.1	K Block Designs . . . . .	37
4.1.2	Uniform Block Design . . . . .	37
4.1.3	Incomplete Designs . . . . .	39
4.2	Balanced Designs . . . . .	39
4.2.1	Sub-types of t-wise balance . . . . .	40
4.2.2	Covering and Packing Designs . . . . .	41
4.3	Constant Replication Design . . . . .	42
4.4	T-designs . . . . .	44
4.5	Steiner Systems . . . . .	45
4.6	Combining block designs . . . . .	45
<b>5</b>	<b>BIBD's</b>	<b>47</b>
5.1	BIBD Basics . . . . .	47
5.2	Necessary Conditions for Existence . . . . .	47
5.2.1	BIBD Param Relationships . . . . .	49
5.3	Constructing New bibd's . . . . .	49
5.3.1	BIBD Complement, Multiple, Combine . . . . .	50
5.3.2	Derived Designs . . . . .	50
5.3.3	Residual Designs . . . . .	51
5.4	Symmetric BIBD's . . . . .	52
5.4.1	Intersection Property on Symmetric BIBDs . . . . .	53
5.4.2	Symmetric BIBD is Simple . . . . .	55
5.4.3	Residual/Derived Sym BIBD Constructions . . . . .	55
5.5	BIBD's and Other Block Designs . . . . .	56
<b>6</b>	<b>Resolvable Designs</b>	<b>57</b>
6.1	Resolutions and Resolution Classes . . . . .	57
6.2	Resolvable Design Locale . . . . .	58
6.3	Resolvable Block Designs . . . . .	59
6.3.1	Bose's Inequality . . . . .	59

<b>7</b>	<b>Group Divisible Designs</b>	<b>60</b>
7.1	Group design . . . . .	60
7.1.1	Group Type . . . . .	61
7.1.2	Uniform Group designs . . . . .	63
7.2	GDD . . . . .	63
7.2.1	Sub types of GDD's . . . . .	65
7.3	GDD and PBD Constructions . . . . .	66
7.3.1	GDD Delete Point construction . . . . .	66
7.3.2	PBD construction from GDD . . . . .	67
7.3.3	Wilson's Construction . . . . .	68
<b>8</b>	<b>Graphs and Designs</b>	<b>70</b>
8.1	Non-empty digraphs . . . . .	70
8.2	Arcs to Blocks . . . . .	71
8.3	Graphs are designs . . . . .	72
8.4	R-regular graphs . . . . .	73
<b>9</b>	<b>Sub-designs</b>	<b>75</b>
9.1	Sub-system and Sub-design Locales . . . . .	75
9.2	Reasoning on Sub-designs . . . . .	77
9.2.1	Reasoning on Incidence Sys property relationships . . . . .	77
9.2.2	Reasoning on Incidence Sys/Design operations . . . . .	78
<b>10</b>	<b>Design Isomorphisms</b>	<b>79</b>
10.1	Images of Set Systems . . . . .	79
10.2	Incidence System Isomorphisms . . . . .	80
10.3	Design Isomorphisms . . . . .	82
10.3.1	Isomorphism Operation . . . . .	82
10.3.2	Design Properties/Operations under Isomorphism . . . . .	83

## 1 Micellaneous Helper Functions on Sets and Multisets

```

theory Multisets-Extras imports
  HOL-Library.Multiset
  Card-Partitions.Set-Partition
  Nested-Multisets-Ordinals.Multiset-More
  Nested-Multisets-Ordinals.Duplicate-Free-Multiset
  HOL-Library.Disjoint-Sets
begin

```

### 1.1 Set Theory Extras

A number of extra helper lemmas for reasoning on sets (finite) required for Design Theory proofs

**lemma** *card-Pow-filter-one:*

**assumes** *finite A*

**shows**  $\text{card } \{x \in \text{Pow } A . \text{card } x = 1\} = \text{card } (A)$

*<proof>*

**lemma** *elem-exists-non-empty-set:*

**assumes**  $\text{card } A > 0$

**obtains** *x where*  $x \in A$

*<proof>*

**lemma** *set-self-imag-compr:*  $\{a \mid a . a \in A\} = A$

*<proof>*

**lemma** *card-subset-not-gt-card:*  $\text{finite } A \implies \text{card } ps > \text{card } A \implies \neg (ps \subseteq A)$

*<proof>*

**lemma** *card-inter-lt-single:*  $\text{finite } A \implies \text{finite } B \implies \text{card } (A \cap B) \leq \text{card } A$

*<proof>*

**lemma** *set-diff-non-empty-not-subset:*

**assumes**  $A \subseteq (B - C)$

**assumes**  $C \neq \{\}$

**assumes**  $A \neq \{\}$

**assumes**  $B \neq \{\}$

**shows**  $\neg (A \subseteq C)$

*<proof>*

**lemma** *set-card-diff-ge-zero:*  $\text{finite } A \implies \text{finite } B \implies A \neq B \implies \text{card } A = \text{card } B \implies$

$\text{card } (A - B) > 0$

*<proof>*

**lemma** *set-filter-diff:*  $\{a \in A . P a\} - \{x\} = \{a \in (A - \{x\}) . (P a)\}$

*<proof>*

**lemma** *set-filter-diff-card:*  $\text{card } (\{a \in A . P a\} - \{x\}) = \text{card } \{a \in (A - \{x\}) . (P a)\}$

*<proof>*

**lemma** *obtain-subset-with-card-int-n:*

**assumes**  $(n :: \text{int}) \leq \text{of-nat } (\text{card } S)$

**assumes**  $(n :: \text{int}) \geq 0$

**obtains** *T where*  $T \subseteq S$   $\text{of-nat } (\text{card } T) = (n :: \text{int})$  *finite T*

*<proof>*

**lemma** *transform-filter-imag-empty-rm:*

**assumes**  $\bigwedge g . g \in G \implies g \neq \{\}$

**shows**  $\{g - \{x\} \mid g . g \in G \wedge g \neq \{x\}\} = \{g - \{x\} \mid g . g \in G\} - \{\{\}\}$

*<proof>*

**lemma** *bij-betw-inter-subsets*:  $\text{bij-betw } f \ A \ B \implies a1 \subseteq A \implies a2 \subseteq A$   
 $\implies f \ ' \ (a1 \cap a2) = (f \ ' \ a1) \cap (f \ ' \ a2)$   
 $\langle \text{proof} \rangle$

Partition related set theory lemmas

**lemma** *partition-on-remove-pt*:  
**assumes** *partition-on*  $A \ G$   
**shows** *partition-on*  $(A - \{x\}) \ \{g - \{x\} \mid g. g \in G \wedge g \neq \{x\}\}$   
 $\langle \text{proof} \rangle$

**lemma** *partition-on-cart-prod*:  
**assumes**  $\text{card } I > 0$   
**assumes**  $A \neq \{\}$   
**assumes**  $G \neq \{\}$   
**assumes** *partition-on*  $A \ G$   
**shows** *partition-on*  $(A \times I) \ \{g \times I \mid g. g \in G\}$   
 $\langle \text{proof} \rangle$

## 1.2 Multiset Helpers

Generic Size, count and card helpers

**lemma** *count-size-set-repr*:  $\text{size } \{\# \ x \in \# \ A \ . \ x = g\#\} = \text{count } A \ g$   
 $\langle \text{proof} \rangle$

**lemma** *mset-nempty-set-nempty*:  $A \neq \{\#\} \iff (\text{set-mset } A) \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *mset-size-ne0-set-card*:  $\text{size } A > 0 \implies \text{card } (\text{set-mset } A) > 0$   
 $\langle \text{proof} \rangle$

**lemma** *set-count-size-min*:  $\text{count } A \ a \geq n \implies \text{size } A \geq n$   
 $\langle \text{proof} \rangle$

**lemma** *card-size-filter-eq*:  $\text{finite } A \implies \text{card } \{a \in A \ . \ P \ a\} = \text{size } \{\# \ a \in \# \ \text{mset-set } A \ . \ P \ a\#\}$   
 $\langle \text{proof} \rangle$

**lemma** *size-multiset-set-mset-const-count*:  
**assumes**  $\text{card } (\text{set-mset } A) = ca$   
**assumes**  $\bigwedge p. p \in \# \ A \implies \text{count } A \ p = ca2$   
**shows**  $\text{size } A = (ca * ca2)$   
 $\langle \text{proof} \rangle$

**lemma** *size-multiset-int-count*:  
**assumes**  $\text{of-nat } (\text{card } (\text{set-mset } A)) = (ca :: \text{int})$   
**assumes**  $\bigwedge p. p \in \# \ A \implies \text{of-nat } (\text{count } A \ p) = (ca2 :: \text{int})$   
**shows**  $\text{of-nat } (\text{size } A) = ((ca :: \text{int}) * ca2)$   
 $\langle \text{proof} \rangle$

**lemma** *mset-union-size*:  $size (A \cup\# B) = size (A) + size (B - A)$   
 ⟨proof⟩

**lemma** *mset-union-size-inter*:  $size (A \cup\# B) = size (A) + size B - size (A \cap\# B)$   
 ⟨proof⟩

Lemmas for repeat\_mset

**lemma** *repeat-mset-size [simp]*:  $size (repeat-mset n A) = n * size A$   
 ⟨proof⟩

**lemma** *repeat-mset-subset-in*:  
**assumes**  $\bigwedge a . a \in\# A \implies a \subseteq B$   
**assumes**  $X \in\# repeat-mset n A$   
**assumes**  $x \in X$   
**shows**  $x \in B$   
 ⟨proof⟩

**lemma** *repeat-mset-not-empty*:  $n > 0 \implies A \neq \{\#\} \implies repeat-mset n A \neq \{\#\}$   
 ⟨proof⟩

**lemma** *elem-in-repeat-in-original*:  $a \in\# repeat-mset n A \implies a \in\# A$   
 ⟨proof⟩

**lemma** *elem-in-original-in-repeat*:  $n > 0 \implies a \in\# A \implies a \in\# repeat-mset n A$   
 ⟨proof⟩

Lemmas on image and filter for multisets

**lemma** *multiset-add-filter-size*:  $size \{\# a \in\# (A1 + A2) . P a \#\} = size \{\# a \in\# A1 . P a \#\} +$   
 $size \{\# a \in\# A2 . P a \#\}$   
 ⟨proof⟩

**lemma** *size-filter-neg*:  $size \{\# a \in\# A . P a \#\} = size A - size \{\# a \in\# A . \neg P a \#\}$   
 ⟨proof⟩

**lemma** *filter-filter-mset-cond-simp*:  
**assumes**  $\bigwedge a . P a \implies Q a$   
**shows**  $filter-mset P A = filter-mset P (filter-mset Q A)$   
 ⟨proof⟩

**lemma** *filter-filter-mset-ss-member*:  $filter-mset (\lambda a . \{x, y\} \subseteq a) A =$   
 $filter-mset (\lambda a . \{x, y\} \subseteq a) (filter-mset (\lambda a . x \in a) A)$   
 ⟨proof⟩

**lemma** *multiset-image-do-nothing*:  $(\bigwedge x . x \in\# A \implies f x = x) \implies image-mset f A = A$

*<proof>*

**lemma** *set-mset-filter*:  $set\text{-}mset \{ \# f a . a \in \# A \# \} = \{ f a \mid a . a \in \# A \}$   
*<proof>*

**lemma** *mset-exists-impl*:  $x \in \# \{ \# f a . a \in \# A \# \} \implies \exists y \in \# A . x = f y$   
*<proof>*

**lemma** *filter-mset-image-mset*:  
 $filter\text{-}mset P (image\text{-}mset f A) = image\text{-}mset f (filter\text{-}mset (\lambda x . P (f x)) A)$   
*<proof>*

**lemma** *mset-bunion-filter*:  $\{ \# a \in \# A . P a \vee Q a \# \} = \{ \# a \in \# A . P a \# \} \cup \# \{ \# a \in \# A . Q a \# \}$   
*<proof>*

**lemma** *mset-inter-filter*:  $\{ \# a \in \# A . P a \wedge Q a \# \} = \{ \# a \in \# A . P a \# \} \cap \# \{ \# a \in \# A . Q a \# \}$   
*<proof>*

**lemma** *image-image-mset*:  $image\text{-}mset (\lambda x . f x) (image\text{-}mset (\lambda y . g y) A) = image\text{-}mset (\lambda x . f (g x)) A$   
*<proof>*

Big Union over multiset helpers

**lemma** *mset-big-union-obtain*:  
**assumes**  $x \in \# \sum \# A$   
**obtains**  $a$  **where**  $a \in \# A$  **and**  $x \in \# a$   
*<proof>*

**lemma** *size-big-union-sum*:  $size (\sum \# (M :: 'a \text{ multiset multiset})) = (\sum x \in \# M . size x)$   
*<proof>*

Cartesian Product on Multisets

**lemma** *size-cartesian-product-singleton* [simp]:  $size (\{ \# a \# \} \times \# B) = size B$   
*<proof>*

**lemma** *size-cartesian-product-singleton-right* [simp]:  $size (A \times \# \{ \# b \# \}) = size A$   
*<proof>*

**lemma** *size-cartesian-product-empty* [simp]:  $size (A \times \# \{ \# \}) = 0$   
*<proof>*

**lemma** *size-add-elem-step-eq*:  
**assumes**  $size (A \times \# B) = size A * size B$   
**shows**  $size (add\text{-}mset x A \times \# B) = size (add\text{-}mset x A) * size B$   
*<proof>*

**lemma** *size-cartesian-product*:  $size (A \times\# B) = size A * size B$   
 ⟨proof⟩

**lemma** *cart-prod-distinct-mset*:  
**assumes** *assm1*: *distinct-mset A*  
**assumes** *assm2*: *distinct-mset B*  
**shows** *distinct-mset (A ×# B)*  
 ⟨proof⟩

**lemma** *cart-product-single-intersect*:  $x1 \neq x2 \implies (\{x1\} \times\# A) \cap\# (\{x2\} \times\# B) = \{\#\}$   
 ⟨proof⟩

**lemma** *size-union-distinct-cart-prod*:  $x1 \neq x2 \implies size ((\{x1\} \times\# A) \cup\# (\{x2\} \times\# B)) = size (\{x1\} \times\# A) + size (\{x2\} \times\# B)$   
 ⟨proof⟩

**lemma** *size-Union-distinct-cart-prod*: *distinct-mset M*  $\implies size (\sum p \in\# M. (\{p\} \times\# B)) = size (M) * size (B)$   
 ⟨proof⟩

**lemma** *size-Union-distinct-cart-prod-filter*: *distinct-mset M*  $\implies (\bigwedge p. p \in\# M \implies size (\{b \in\# B. P p b\}) = c) \implies size (\sum p \in\# M. (\{p\} \times\# \{b \in\# B. P p b\})) = size (M) * c$   
 ⟨proof⟩

**lemma** *size-Union-distinct-cart-prod-filter2*: *distinct-mset V*  $\implies (\bigwedge b. b \in\# B \implies size (\{v \in\# V. P v b\}) = c) \implies size (\sum b \in\# B. (\{b\} \times\# \{v \in\# V. P v b\})) = size (B) * c$   
 ⟨proof⟩

**lemma** *cart-product-add-1*:  $(add-mset a A) \times\# B = (\{a\} \times\# B) + (A \times\# B)$   
 ⟨proof⟩

**lemma** *cart-product-add-1-filter*:  $\{m \in\# ((add-mset a M) \times\# N). P m\} = \{m \in\# (M \times\# N). P m\} + \{m \in\# (\{a\} \times\# N). P m\}$   
 ⟨proof⟩

**lemma** *cart-product-add-1-filter2*:  $\{m \in\# (M \times\# (add-mset b N)). P m\} = \{m \in\# (M \times\# N). P m\} + \{m \in\# (M \times\# \{b\}). P m\}$   
 ⟨proof⟩

**lemma** *cart-prod-singleton-right-gen*:  
**assumes**  $\bigwedge x. x \in\# (A \times\# \{b\}) \implies P x \longleftrightarrow Q (fst x)$   
**shows**  $\{x \in\# (A \times\# \{b\}). P x\} = \{a \in\# A. Q a\} \times\# \{b\}$   
 ⟨proof⟩



**lemma** *cart-prod-singleton-left-gen*:

**assumes**  $\bigwedge x . x \in \# (\{ \#a \} \times \# B) \implies P x \longleftrightarrow Q (snd x)$

**shows**  $\{ \#x \in \# (\{ \#a \} \times \# B) . P x \# \} = \{ \#a \} \times \# \{ \#b \in \# B . Q b \# \}$

*<proof>*

**lemma** *cart-product-singleton-left*:  $\{ \#m \in \# (\{ \#a \} \times \# N) . fst m \in snd m \# \}$

$=$

$(\{ \#a \} \times \# \{ \# n \in \# N . a \in n \# \})$  (**is**  $?A = ?B$ )

*<proof>*

**lemma** *cart-product-singleton-right*:  $\{ \#m \in \# (N \times \# \{ \#b \}) . fst m \in snd m \# \} =$

$(\{ \# n \in \# N . n \in b \# \} \times \# \{ \# b \})$  (**is**  $?A = ?B$ )

*<proof>*

**lemma** *cart-product-add-1-filter-eq*:  $\{ \#m \in \# ((add-mset a M) \times \# N) . (fst m \in snd m) \# \} =$

$\{ \#m \in \# (M \times \# N) . (fst m \in snd m) \# \} + (\{ \#a \} \times \# \{ \# n \in \# N . a \in n \# \})$

*<proof>*

**lemma** *cart-product-add-1-filter-eq-mirror*:  $\{ \#m \in \# M \times \# (add-mset b N) . (fst m \in snd m) \# \} =$

$\{ \#m \in \# (M \times \# N) . (fst m \in snd m) \# \} + (\{ \# n \in \# M . n \in b \# \} \times \# \{ \#b \})$

*<proof>*

**lemma** *set-break-down-left*:

**shows**  $\{ \# m \in \# (M \times \# N) . (fst m) \in (snd m) \# \} = (\sum m \in \# M . (\{ \#m \# \} \times \# \{ \#n \in \# N . m \in n \# \}))$

*<proof>*

**lemma** *set-break-down-right*:

**shows**  $\{ \# x \in \# M \times \# N . (fst x) \in (snd x) \# \} = (\sum n \in \# N . (\{ \#m \in \# M . m \in n \# \} \times \# \{ \#n \# \}))$

*<proof>*

Reasoning on sums of elements over multisets

**lemma** *sum-over-fun-eq*:

**assumes**  $\bigwedge x . x \in \# A \implies f x = g x$

**shows**  $(\sum x \in \# A . f(x)) = (\sum x \in \# A . g(x))$

*<proof>*

**lemma** *sum-mset-add-diff-nat*:

**fixes**  $x :: 'a$  **and**  $f g :: 'a \Rightarrow nat$

**assumes**  $\bigwedge x . x \in \# A \implies f x \geq g x$

**shows**  $(\sum x \in \# A . f x - g x) = (\sum x \in \# A . f x) - (\sum x \in \# A . g x)$

*<proof>*

**lemma** *sum-mset-add-diff-int*:

**fixes**  $x :: 'a$  **and**  $f g :: 'a \Rightarrow \text{int}$

**shows**  $(\sum x \in\# A. f x - g x) = (\sum x \in\# A. f x) - (\sum x \in\# A. g x)$

*<proof>*

**context** *ring-1*

**begin**

**lemma** *sum-mset-add-diff*:  $(\sum x \in\# A. f x - g x) = (\sum x \in\# A. f x) - (\sum x \in\# A. g x)$

*<proof>*

**end**

**context** *ordered-semiring*

**begin**

**lemma** *sum-mset-ge0*:  $(\bigwedge x. f x \geq 0) \implies (\sum x \in\# A. f x) \geq 0$

*<proof>*

**lemma** *sum-order-add-mset*:  $(\bigwedge x. f x \geq 0) \implies (\sum x \in\# A. f x) \leq (\sum x \in\# A. f x)$

*<proof>*

**lemma** *sum-mset-0-left*:  $(\bigwedge x. f x \geq 0) \implies (\sum x \in\# A. f x) = 0 \implies (\forall x \in\# A. f x = 0)$

*<proof>*

**lemma** *sum-mset-0-iff-ge-0*:

**assumes**  $(\bigwedge x. f x \geq 0)$

**shows**  $(\sum x \in\# A. f x) = 0 \iff (\forall x \in \text{set-mset } A. f x = 0)$

*<proof>*

**end**

**lemma** *mset-set-size-card-count*:  $(\sum x \in\# A. x) = (\sum x \in \text{set-mset } A. x * (\text{count } A x))$

*<proof>*

### 1.3 Partitions on Multisets

A partition on a multiset  $A$  is a multiset of multisets, where the sum over  $P$  equals  $A$  and the empty multiset is not in the partition. Based off set partition definition. We note that unlike set partitions, there is no requirement for elements in the multisets to be distinct due to the definition of union on multisets [1]

**lemma** *mset-size-partition-dep*:  $\text{size } \{\# a \in\# A. P a \vee Q a \#\} =$

$size \{ \# a \in \# A . P a \# \} + size \{ \# a \in \# A . Q a \# \} - size \{ \# a \in \# A . P a \wedge Q a \# \}$   
 <proof>

**definition** *partition-on-mset* :: 'a multiset  $\Rightarrow$  'a multiset multiset  $\Rightarrow$  bool **where**  
*partition-on-mset* A P  $\longleftrightarrow \sum \#P = A \wedge \{ \# \} \notin \# P$

**lemma** *partition-on-msetI* [intro]:  $\sum \#P = A \Longrightarrow \{ \# \} \notin \# P \Longrightarrow$  *partition-on-mset*  
 A P  
 <proof>

**lemma** *partition-on-msetD1*: *partition-on-mset* A P  $\Longrightarrow \sum \#P = A$   
 <proof>

**lemma** *partition-on-msetD2*: *partition-on-mset* A P  $\Longrightarrow \{ \# \} \notin \# P$   
 <proof>

**lemma** *partition-on-mset-empty*: *partition-on-mset*  $\{ \# \}$  P  $\longleftrightarrow P = \{ \# \}$   
 <proof>

**lemma** *partition-on-mset-all*:  $A \neq \{ \# \} \Longrightarrow$  *partition-on-mset* A  $\{ \# A \# \}$   
 <proof>

**lemma** *partition-on-mset-singletons*: *partition-on-mset* A (image-mset  $(\lambda x . \{ \# x \# \})$   
 A)  
 <proof>

**lemma** *partition-on-mset-not-empty*:  $A \neq \{ \# \} \Longrightarrow$  *partition-on-mset* A P  $\Longrightarrow P$   
 $\neq \{ \# \}$   
 <proof>

**lemma** *partition-on-msetI2*:  $\sum \#P = A \Longrightarrow (\bigwedge p . p \in \# P \Longrightarrow p \neq \{ \# \}) \Longrightarrow$   
*partition-on-mset* A P  
 <proof>

**lemma** *partition-on-mset-elems*: *partition-on-mset* A P  $\Longrightarrow p1 \in \# P \Longrightarrow x \in \#$   
 $p1 \Longrightarrow x \in \# A$   
 <proof>

**lemma** *partition-on-mset-sum-size-eq*: *partition-on-mset* A P  $\Longrightarrow (\sum x \in \# P . size$   
 $x) = size A$   
 <proof>

**lemma** *partition-on-mset-card*: **assumes** *partition-on-mset* A P **shows**  $size P \leq$   
 $size A$   
 <proof>

**lemma** *partition-on-mset-count-eq*: *partition-on-mset* A P  $\Longrightarrow a \in \# A \Longrightarrow$   
 $(\sum x \in \# P . count x a) = count A a$

*<proof>*

**lemma** *partition-on-mset-subsets*:  $\text{partition-on-mset } A \ P \implies x \in\# \ P \implies x \subseteq\# \ A$   
*<proof>*

**lemma** *partition-on-mset-distinct*:  
**assumes** *partition-on-mset*  $A \ P$   
**assumes** *distinct-mset*  $A$   
**shows** *distinct-mset*  $P$   
*<proof>*

**lemma** *partition-on-mset-distinct-disjoint*:  
**assumes** *partition-on-mset*  $A \ P$   
**assumes** *distinct-mset*  $A$   
**assumes**  $p1 \in\# \ P$   
**assumes**  $p2 \in\# \ P - \{\#p1\# \}$   
**shows**  $p1 \cap\# \ p2 = \{\# \}$   
*<proof>*

**lemma** *partition-on-mset-diff*:  
**assumes** *partition-on-mset*  $A \ P$   
**assumes**  $Q \subseteq\# \ P$   
**shows** *partition-on-mset*  $(A - \sum\# \ Q) \ (P - Q)$   
*<proof>*

**lemma** *sigma-over-set-partition-count*:  
**assumes** *finite*  $A$   
**assumes** *partition-on*  $A \ P$   
**assumes**  $x \in\# \ \sum\# \ (\text{mset-set } (\text{mset-set } ' P))$   
**shows**  $\text{count } (\sum\# \ (\text{mset-set } (\text{mset-set } ' P))) \ x = 1$   
*<proof>*

**lemma** *partition-on-mset-set*:  
**assumes** *finite*  $A$   
**assumes** *partition-on*  $A \ P$   
**shows** *partition-on-mset*  $(\text{mset-set } A) \ (\text{mset-set } (\text{image } (\lambda \ x. \ \text{mset-set } x) \ P))$   
*<proof>*

**lemma** *partition-on-mset-distinct-inter*:  
**assumes** *partition-on-mset*  $A \ P$   
**assumes** *distinct-mset*  $A$   
**assumes**  $p1 \in\# \ P$  **and**  $p2 \in\# \ P$  **and**  $p1 \neq p2$   
**shows**  $p1 \cap\# \ p2 = \{\# \}$   
*<proof>*

**lemma** *partition-on-set-mset-distinct*:  
**assumes** *partition-on-mset*  $A \ P$   
**assumes** *distinct-mset*  $A$

**assumes**  $p \in\# \text{ image-mset set-mset } P$   
**assumes**  $p' \in\# \text{ image-mset set-mset } P$   
**assumes**  $p \neq p'$   
**shows**  $p \cap p' = \{\}$   
 <proof>

**lemma** *partition-on-set-mset*:  
**assumes** *partition-on-mset*  $A P$   
**assumes** *distinct-mset*  $A$   
**shows** *partition-on* (*set-mset*  $A$ ) (*set-mset* (*image-mset set-mset*  $P$ ))  
 <proof>

**lemma** *partition-on-mset-eq-imp-eq-carrier*:  
**assumes** *partition-on-mset*  $A P$   
**assumes** *partition-on-mset*  $B P$   
**shows**  $A = B$   
 <proof>

**lemma** *partition-on-mset-add-single*:  
**assumes** *partition-on-mset*  $A P$   
**shows** *partition-on-mset* (*add-mset*  $a A$ ) (*add-mset*  $\{ \#a\# \} P$ )  
 <proof>

**lemma** *partition-on-mset-add-part*:  
**assumes** *partition-on-mset*  $A P$   
**assumes**  $X \neq \{\#\}$   
**assumes**  $A + X = A'$   
**shows** *partition-on-mset*  $A'$  (*add-mset*  $X P$ )  
 <proof>

**lemma** *partition-on-mset-add*:  
**assumes** *partition-on-mset*  $A P$   
**assumes**  $X \in\# P$   
**assumes** *add-mset*  $a X = X'$   
**shows** *partition-on-mset* (*add-mset*  $a A$ ) (*add-mset*  $X' (P - \{\#X\#})$ )  
 <proof>

**lemma** *partition-on-mset-elem-exists-part*:  
**assumes** *partition-on-mset*  $A P$   
**assumes**  $x \in\# A$   
**obtains**  $p$  **where**  $p \in\# P$  **and**  $x \in\# p$   
 <proof>

**lemma** *partition-on-mset-combine*:  
**assumes** *partition-on-mset*  $A P$   
**assumes** *partition-on-mset*  $B Q$   
**shows** *partition-on-mset*  $(A + B) (P + Q)$   
 <proof>

```

lemma partition-on-mset-split:
  assumes partition-on-mset  $A (P + Q)$ 
  shows partition-on-mset  $(\sum \#P) P$ 
   $\langle proof \rangle$ 
end
theory Design-Basics imports Main Multisets-Extras HOL-Library.Disjoint-Sets
begin

```

## 2 Design Theory Basics

All definitions in this section reference the handbook of combinatorial designs [3]

### 2.1 Initial setup

Enable coercion of nats to ints to aid with reasoning on design properties

```

declare  $[[coercion-enabled]]$ 
declare  $[[coercion\ of\ nat :: nat \Rightarrow int]]$ 

```

### 2.2 Incidence System

An incidence system is defined to be a wellformed set system. i.e. each block is a subset of the base point set. Alternatively, an incidence system can be looked at as the point set and an incidence relation which indicates if they are in the same block

```

locale incidence-system =
  fixes point-set :: 'a set ( $\mathcal{V}$ )
  fixes block-collection :: 'a set multiset ( $\mathcal{B}$ )
  assumes wellformed:  $b \in \# \mathcal{B} \Longrightarrow b \subseteq \mathcal{V}$ 
begin

```

```

definition  $\mathcal{I} \equiv \{ (x, b) . b \in \# \mathcal{B} \wedge x \in b \}$ 

```

```

definition incident :: 'a  $\Rightarrow$  'a set  $\Rightarrow$  bool where
incident  $p\ b \equiv (p, b) \in \mathcal{I}$ 

```

Defines common notation used to indicate number of points ( $v$ ) and number of blocks ( $b$ )

```

abbreviation  $v \equiv card\ \mathcal{V}$ 

```

```

abbreviation  $b \equiv size\ \mathcal{B}$ 

```

Basic incidence lemmas

```

lemma incidence-alt-def:
  assumes  $p \in \mathcal{V}$ 
  assumes  $b \in \# \mathcal{B}$ 

```

**shows** *incident*  $p\ b \longleftrightarrow p \in b$   
*<proof>*

**lemma** *wf-invalid-point*:  $x \notin \mathcal{V} \implies b \in \# \mathcal{B} \implies x \notin b$   
*<proof>*

**lemma** *block-set-nempty-imp-block-ex*:  $\mathcal{B} \neq \{\#\} \implies \exists\ bl.\ bl \in \# \mathcal{B}$   
*<proof>*

Abbreviations for all incidence systems

**abbreviation** *multiplicity* :: 'a set  $\Rightarrow$  nat **where**  
*multiplicity*  $b \equiv \text{count } \mathcal{B}\ b$

**abbreviation** *incomplete-block* :: 'a set  $\Rightarrow$  bool **where**  
*incomplete-block*  $bl \equiv \text{card } bl < \text{card } \mathcal{V} \wedge bl \in \# \mathcal{B}$

**lemma** *incomplete-alt-size*: *incomplete-block*  $bl \implies \text{card } bl < v$   
*<proof>*

**lemma** *incomplete-alt-in*: *incomplete-block*  $bl \implies bl \in \# \mathcal{B}$   
*<proof>*

**lemma** *incomplete-alt-imp[intro]*:  $\text{card } bl < v \implies bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
*<proof>*

**definition** *design-support* :: 'a set set **where**  
*design-support*  $\equiv \text{set-mset } \mathcal{B}$

**end**

## 2.3 Finite Incidence Systems

These simply require the point set to be finite. As multisets are only defined to be finite, it is implied that the block set must be finite already

**locale** *finite-incidence-system* = *incidence-system* +  
**assumes** *finite-sets*: *finite*  $\mathcal{V}$   
**begin**

**lemma** *finite-blocks*:  $b \in \# \mathcal{B} \implies \text{finite } b$   
*<proof>*

**lemma** *mset-points-distinct*: *distinct-mset* (*mset-set*  $\mathcal{V}$ )  
*<proof>*

**lemma** *mset-points-distinct-diff-one*: *distinct-mset* (*mset-set* ( $\mathcal{V} - \{x\}$ ))  
*<proof>*

**lemma** *finite-design-support*: *finite* (*design-support*)

*<proof>*

**lemma** *block-size-lt-order*:  $bl \in \# \mathcal{B} \implies \text{card } bl \leq \text{card } \mathcal{V}$   
*<proof>*

**end**

## 2.4 Designs

There are many varied definitions of a design in literature. However, the most commonly accepted definition is a finite point set,  $V$  and collection of blocks  $B$ , where no block in  $B$  can be empty

**locale** *design* = *finite-incidence-system* +  
**assumes** *blocks-nonempty*:  $bl \in \# \mathcal{B} \implies bl \neq \{\}$   
**begin**

**lemma** *wf-design*: *design*  $\mathcal{V} \mathcal{B}$  *<proof>*

**lemma** *wf-design-iff*:  $bl \in \# \mathcal{B} \implies \text{design } \mathcal{V} \mathcal{B} \longleftrightarrow (bl \subseteq \mathcal{V} \wedge \text{finite } \mathcal{V} \wedge bl \neq \{\})$   
*<proof>*

Reasoning on non empty properties and non zero parameters

**lemma** *blocks-nonempty-alt*:  $\forall bl \in \# \mathcal{B}. bl \neq \{\}$   
*<proof>*

**lemma** *block-set-nonempty-imp-points*:  $\mathcal{B} \neq \{\#\} \implies \mathcal{V} \neq \{\}$   
*<proof>*

**lemma** *b-non-zero-imp-v-non-zero*:  $b > 0 \implies v > 0$   
*<proof>*

**lemma** *v-eq0-imp-b-eq0*:  $v = 0 \implies b = 0$   
*<proof>*

Size lemmas

**lemma** *block-size-lt-v*:  $bl \in \# \mathcal{B} \implies \text{card } bl \leq v$   
*<proof>*

**lemma** *block-size-gt-0*:  $bl \in \# \mathcal{B} \implies \text{card } bl > 0$   
*<proof>*

**lemma** *design-cart-product-size*:  $\text{size } ((\text{mset-set } \mathcal{V}) \times \# \mathcal{B}) = v * b$   
*<proof>*

**end**

Intro rules for design locale

**lemma** *wf-design-implies*:  
**assumes**  $(\bigwedge b. b \in \# \mathcal{B} \implies b \subseteq V)$



**assumes**  $\bigwedge b. b \in \# \mathcal{B} \implies b \neq \{\}$   
**assumes** *finite*  $\mathcal{V}$   
**assumes**  $\mathcal{B} \neq \{\#\}$   
**assumes**  $\mathcal{V} \neq \{\}$   
**shows** *design*  $\mathcal{V} \mathcal{B}$   
 $\langle$ *proof* $\rangle$

**lemma** (*in incidence-system*) *finite-sysI*[*intro*]: *finite*  $\mathcal{V} \implies$  *finite-incidence-system*  $\mathcal{V} \mathcal{B}$   
 $\langle$ *proof* $\rangle$

**lemma** (*in finite-incidence-system*) *designI*[*intro*]:  $(\bigwedge b. b \in \# \mathcal{B} \implies b \neq \{\}) \implies$   
 $\mathcal{B} \neq \{\#\}$   
 $\implies \mathcal{V} \neq \{\} \implies$  *design*  $\mathcal{V} \mathcal{B}$   
 $\langle$ *proof* $\rangle$

## 2.5 Core Property Definitions

### 2.5.1 Replication Number

The replication number for a point is the number of blocks that point is incident with

**definition** *point-replication-number* :: '*a set multiset*  $\Rightarrow$  '*a*  $\Rightarrow$  *nat* (**infix** *rep* 75)  
**where**  
 $B \text{ rep } x \equiv \text{size } \{\#b \in \# B . x \in b\# \}$

**lemma** *max-point-rep*:  $B \text{ rep } x \leq \text{size } B$   
 $\langle$ *proof* $\rangle$

**lemma** *rep-number-g0-exists*:  
**assumes**  $B \text{ rep } x > 0$   
**obtains**  $b$  **where**  $b \in \# B$  **and**  $x \in b$   
 $\langle$ *proof* $\rangle$

**lemma** *rep-number-on-set-def*: *finite*  $B \implies$   $(\text{mset-set } B) \text{ rep } x = \text{card } \{b \in B . x \in b\}$   
 $\langle$ *proof* $\rangle$

**lemma** *point-rep-number-split*[*simp*]:  $(A + B) \text{ rep } x = A \text{ rep } x + B \text{ rep } x$   
 $\langle$ *proof* $\rangle$

**lemma** *point-rep-singleton-val* [*simp*]:  $x \in b \implies \{\#b\# \} \text{ rep } x = 1$   
 $\langle$ *proof* $\rangle$

**lemma** *point-rep-singleton-ival* [*simp*]:  $x \notin b \implies \{\#b\# \} \text{ rep } x = 0$   
 $\langle$ *proof* $\rangle$

**context** *incidence-system*  
**begin**

**lemma** *point-rep-number-alt-def*:  $\mathcal{B} \text{ rep } x = \text{size } \{\# b \in \# \mathcal{B} . x \in b\# \}$   
 ⟨proof⟩

**lemma** *rep-number-non-zero-system-point*:  $\mathcal{B} \text{ rep } x > 0 \implies x \in \mathcal{V}$   
 ⟨proof⟩

**lemma** *point-rep-non-existence [simp]*:  $x \notin \mathcal{V} \implies \mathcal{B} \text{ rep } x = 0$   
 ⟨proof⟩

**lemma** *point-rep-number-inv*:  $\text{size } \{\# b \in \# \mathcal{B} . x \notin b\# \} = b - (\mathcal{B} \text{ rep } x)$   
 ⟨proof⟩

**lemma** *point-rep-num-inv-non-empty*:  $(\mathcal{B} \text{ rep } x) < b \implies \mathcal{B} \neq \{\#\} \implies \{\# b \in \# \mathcal{B} . x \notin b\# \} \neq \{\#\}$   
 ⟨proof⟩

**end**

## 2.5.2 Point Index

The point index of a subset of points in a design, is the number of times those points occur together in a block of the design

**definition** *points-index* :: 'a set multiset  $\Rightarrow$  'a set  $\Rightarrow$  nat (**infix** *index* 75) **where**  
 $B \text{ index } ps \equiv \text{size } \{\# b \in \# B . ps \subseteq b\# \}$

**lemma** *points-index-empty [simp]*:  $\{\#\} \text{ index } ps = 0$   
 ⟨proof⟩

**lemma** *point-index-distrib*:  $(B1 + B2) \text{ index } ps = B1 \text{ index } ps + B2 \text{ index } ps$   
 ⟨proof⟩

**lemma** *point-index-diff*:  $B1 \text{ index } ps = (B1 + B2) \text{ index } ps - B2 \text{ index } ps$   
 ⟨proof⟩

**lemma** *points-index-singleton*:  $\{\#b\# \} \text{ index } ps = 1 \iff ps \subseteq b$   
 ⟨proof⟩

**lemma** *points-index-singleton-zero*:  $\neg (ps \subseteq b) \implies \{\#b\# \} \text{ index } ps = 0$   
 ⟨proof⟩

**lemma** *points-index-sum*:  $(\sum \# B) \text{ index } ps = (\sum b \in \# B . (b \text{ index } ps))$   
 ⟨proof⟩

**lemma** *points-index-block-image-add-eq*:  
**assumes**  $x \notin ps$   
**assumes**  $B \text{ index } ps = l$   
**shows**  $\{\# \text{ insert } x b . b \in \# B\# \} \text{ index } ps = l$   
 ⟨proof⟩

**lemma** *points-index-on-set-def* [*simp*]:

**assumes** *finite B*

**shows**  $(\text{mset-set } B) \text{ index } ps = \text{card } \{b \in B. ps \subseteq b\}$

*<proof>*

**lemma** *points-index-single-rep-num*:  $B \text{ index } \{x\} = B \text{ rep } x$

*<proof>*

**lemma** *points-index-pair-rep-num*:

**assumes**  $\bigwedge b. b \in \# B \implies x \in b$

**shows**  $B \text{ index } \{x, y\} = B \text{ rep } y$

*<proof>*

**lemma** *points-index-0-left-imp*:

**assumes**  $B \text{ index } ps = 0$

**assumes**  $b \in \# B$

**shows**  $\neg (ps \subseteq b)$

*<proof>*

**lemma** *points-index-0-right-imp*:

**assumes**  $\bigwedge b. b \in \# B \implies (\neg ps \subseteq b)$

**shows**  $B \text{ index } ps = 0$

*<proof>*

**lemma** *points-index-0-iff*:  $B \text{ index } ps = 0 \iff (\forall b. b \in \# B \longrightarrow (\neg ps \subseteq b))$

*<proof>*

**lemma** *points-index-gt0-impl-existance*:

**assumes**  $B \text{ index } ps > 0$

**shows**  $(\exists bl. (bl \in \# B \wedge ps \subseteq bl))$

*<proof>*

**lemma** *points-index-one-unique*:

**assumes**  $B \text{ index } ps = 1$

**assumes**  $bl \in \# B$  **and**  $ps \subseteq bl$  **and**  $bl' \in \# B$  **and**  $ps \subseteq bl'$

**shows**  $bl = bl'$

*<proof>*

**lemma** *points-index-one-unique-block*:

**assumes**  $B \text{ index } ps = 1$

**shows**  $\exists! bl. (bl \in \# B \wedge ps \subseteq bl)$

*<proof>*

**lemma** *points-index-one-not-unique-block*:

**assumes**  $B \text{ index } ps = 1$

**assumes**  $ps \subseteq bl$

**assumes**  $bl \in \# B$

**assumes**  $bl' \in \# B - \{\#bl\# \}$

**shows**  $\neg ps \subseteq bl'$   
 ⟨proof⟩

**lemma** (in *incidence-system*) *points-index-alt-def*:  $\mathcal{B} \text{ index } ps = \text{size } \{\#b \in \# \mathcal{B} . ps \subseteq b\# \}$   
 ⟨proof⟩

**lemma** (in *incidence-system*) *points-index-ps-nin*:  $\neg (ps \subseteq \mathcal{V}) \implies \mathcal{B} \text{ index } ps = 0$   
 ⟨proof⟩

**lemma** (in *incidence-system*) *points-index-count-bl*:  
*multiplicity*  $bl \geq n \implies ps \subseteq bl \implies \text{count } \{\#bl \in \# \mathcal{B} . ps \subseteq bl\# \} bl \geq n$   
 ⟨proof⟩

**lemma** (in *finite-incidence-system*) *points-index-zero*:  
**assumes**  $\text{card } ps > \text{card } \mathcal{V}$   
**shows**  $\mathcal{B} \text{ index } ps = 0$   
 ⟨proof⟩

**lemma** (in *design*) *points-index-subset*:  
 $x \subseteq \# \{\#bl \in \# \mathcal{B} . ps \subseteq bl\# \} \implies ps \subseteq \mathcal{V} \implies (\mathcal{B} \text{ index } ps) \geq (\text{size } x)$   
 ⟨proof⟩

**lemma** (in *design*) *points-index-count-min*: *multiplicity*  $bl \geq n \implies ps \subseteq bl \implies \mathcal{B} \text{ index } ps \geq n$   
 ⟨proof⟩

### 2.5.3 Intersection Number

The intersection number of two blocks is the size of the intersection of those blocks. i.e. the number of points which occur in both blocks

**definition** *intersection-number* :: 'a set  $\Rightarrow$  'a set  $\Rightarrow$  nat (infix  $|\cap|$  70) **where**  
 $b1 |\cap| b2 \equiv \text{card } (b1 \cap b2)$

**lemma** *intersection-num-non-neg*:  $b1 |\cap| b2 \geq 0$   
 ⟨proof⟩

**lemma** *intersection-number-empty-iff*:  
**assumes** *finite*  $b1$   
**shows**  $b1 \cap b2 = \{\} \longleftrightarrow b1 |\cap| b2 = 0$   
 ⟨proof⟩

**lemma** *intersect-num-commute*:  $b1 |\cap| b2 = b2 |\cap| b1$   
 ⟨proof⟩

**definition** *n-intersect-number* :: 'a set  $\Rightarrow$  nat  $\Rightarrow$  'a set  $\Rightarrow$  nat **where**  
*n-intersect-number*  $b1 \ n \ b2 \equiv \text{card } \{ x \in \text{Pow } (b1 \cap b2) . \text{card } x = n \}$

**notation** *n-intersect-number* ((-  $|\cap|$  -) [52, 51, 52] 50)

**lemma** *n-intersect-num-subset-def*:  $b1 \mid\cap\mid_n b2 = \text{card } \{x . x \subseteq b1 \cap b2 \wedge \text{card } x = n\}$   
 ⟨proof⟩

**lemma** *n-inter-num-one*:  $\text{finite } b1 \implies \text{finite } b2 \implies b1 \mid\cap\mid_1 b2 = b1 \mid\cap\mid b2$   
 ⟨proof⟩

**lemma** *n-inter-num-choose*:  $\text{finite } b1 \implies \text{finite } b2 \implies b1 \mid\cap\mid_n b2 = (\text{card } (b1 \cap b2) \text{ choose } n)$   
 ⟨proof⟩

**lemma** *set-filter-single*:  $x \in A \implies \{a \in A . a = x\} = \{x\}$   
 ⟨proof⟩

**lemma** (in *design*) *n-inter-num-zero*:  
 assumes  $b1 \in\# \mathcal{B}$  and  $b2 \in\# \mathcal{B}$   
 shows  $b1 \mid\cap\mid_0 b2 = 1$   
 ⟨proof⟩

**lemma** (in *design*) *n-inter-num-choose-design*:  $b1 \in\# \mathcal{B} \implies b2 \in\# \mathcal{B} \implies b1 \mid\cap\mid_n b2 = (\text{card } (b1 \cap b2) \text{ choose } n)$   
 ⟨proof⟩

**lemma** (in *design*) *n-inter-num-choose-design-inter*:  $b1 \in\# \mathcal{B} \implies b2 \in\# \mathcal{B} \implies b1 \mid\cap\mid_n b2 = (\text{nat } (b1 \mid\cap\mid b2) \text{ choose } n)$   
 ⟨proof⟩

## 2.6 Incidence System Set Property Definitions

**context** *incidence-system*

**begin**

The set of replication numbers for all points of design

**definition** *replication-numbers* :: *nat set* **where**  
 $\text{replication-numbers} \equiv \{\mathcal{B} \text{ rep } x \mid x . x \in \mathcal{V}\}$

**lemma** *replication-numbers-non-empty*:  
 assumes  $\mathcal{V} \neq \{\}$   
 shows  $\text{replication-numbers} \neq \{\}$   
 ⟨proof⟩

**lemma** *obtain-point-with-rep*:  $r \in \text{replication-numbers} \implies \exists x . x \in \mathcal{V} \wedge \mathcal{B} \text{ rep } x = r$   
 ⟨proof⟩

**lemma** *point-rep-number-in-set*:  $x \in \mathcal{V} \implies (\mathcal{B} \text{ rep } x) \in \text{replication-numbers}$   
 ⟨proof⟩

**lemma** (in *finite-incidence-system*) *replication-numbers-finite*: *finite replication-numbers*

*<proof>*

The set of all block sizes in a system

**definition** *sys-block-sizes* :: *nat set* **where**  
*sys-block-sizes*  $\equiv \{ \text{card } bl \mid bl. bl \in \# \mathcal{B} \}$

**lemma** *block-sizes-non-empty-set*:

**assumes**  $\mathcal{B} \neq \{\#\}$

**shows** *sys-block-sizes*  $\neq \{\}$

*<proof>*

**lemma** *finite-block-sizes*: *finite* (*sys-block-sizes*)

*<proof>*

**lemma** *block-sizes-non-empty*:

**assumes**  $\mathcal{B} \neq \{\#\}$

**shows** *card* (*sys-block-sizes*)  $> 0$

*<proof>*

**lemma** *sys-block-sizes-in*:  $bl \in \# \mathcal{B} \implies \text{card } bl \in \text{sys-block-sizes}$

*<proof>*

**lemma** *sys-block-sizes-obtain-bl*:  $x \in \text{sys-block-sizes} \implies (\exists bl \in \# \mathcal{B}. \text{card } bl = x)$

*<proof>*

The set of all possible intersection numbers in a system.

**definition** *intersection-numbers* :: *nat set* **where**

*intersection-numbers*  $\equiv \{ b1 \mid \cap \mid b2 \mid b1 \ b2 . b1 \in \# \mathcal{B} \wedge b2 \in \# (\mathcal{B} - \{\#b1\#\}) \}$

**lemma** *obtain-blocks-intersect-num*:  $n \in \text{intersection-numbers} \implies$

$\exists b1 \ b2. b1 \in \# \mathcal{B} \wedge b2 \in \# (\mathcal{B} - \{\#b1\#\}) \wedge b1 \mid \cap \mid b2 = n$

*<proof>*

**lemma** *intersect-num-in-set*:  $b1 \in \# \mathcal{B} \implies b2 \in \# (\mathcal{B} - \{\#b1\#\}) \implies b1 \mid \cap \mid b2 \in \text{intersection-numbers}$

*<proof>*

The set of all possible point indices

**definition** *point-indices* :: *nat*  $\Rightarrow$  *nat set* **where**

*point-indices*  $t \equiv \{ \mathcal{B} \ \text{index } ps \mid ps. \text{card } ps = t \wedge ps \subseteq \mathcal{V} \}$

**lemma** *point-indices-elem-in*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \ \text{index } ps \in \text{point-indices } t$

*<proof>*

**lemma** *point-indices-alt-def*:  $\text{point-indices } t = \{ \mathcal{B} \ \text{index } ps \mid ps. \text{card } ps = t \wedge ps \subseteq \mathcal{V} \}$

*<proof>*

**end**

## 2.7 Basic Constructions on designs

This section defines some of the most common universal constructions found in design theory involving only a single design

### 2.7.1 Design Complements

**context** *incidence-system*  
**begin**

The complement of a block are all the points in the design not in that block. The complement of a design is therefore the original point sets, and set of all block complements

**definition** *block-complement*:: 'a set  $\Rightarrow$  'a set ( $-^c$  [56] 55) **where**  
*block-complement*  $b \equiv \mathcal{V} - b$

**definition** *complement-blocks* :: 'a set multiset ( $(\mathcal{B}^C)$ ) **where**  
*complement-blocks*  $\equiv \{\# \text{ } bl^c . bl \in \# \mathcal{B} \#\}$

**lemma** *block-complement-elem-iff*:  
**assumes**  $ps \subseteq \mathcal{V}$   
**shows**  $ps \subseteq bl^c \longleftrightarrow (\forall x \in ps. x \notin bl)$   
*<proof>*

**lemma** *block-complement-inter-empty*:  $bl1^c = bl2 \implies bl1 \cap bl2 = \{\}$   
*<proof>*

**lemma** *block-complement-inv*:  
**assumes**  $bl \in \# \mathcal{B}$   
**assumes**  $bl^c = bl2$   
**shows**  $bl2^c = bl$   
*<proof>*

**lemma** *block-complement-subset-points*:  $ps \subseteq (bl^c) \implies ps \subseteq \mathcal{V}$   
*<proof>*

**lemma** *obtain-comp-block-orig*:  
**assumes**  $bl1 \in \# \mathcal{B}^C$   
**obtains**  $bl2$  **where**  $bl2 \in \# \mathcal{B}$  **and**  $bl1 = bl2^c$   
*<proof>*

**lemma** *complement-same-b [simp]*:  $size \mathcal{B}^C = size \mathcal{B}$   
*<proof>*

**lemma** *block-comp-elem-alt-left*:  $x \in bl \implies ps \subseteq bl^c \implies x \notin ps$   
*<proof>*

**lemma** *block-comp-elem-alt-right*:  $ps \subseteq \mathcal{V} \implies (\bigwedge x . x \in ps \implies x \notin bl) \implies ps \subseteq bl^c$

*<proof>*

**lemma** *complement-index:*

**assumes**  $ps \subseteq \mathcal{V}$

**shows**  $\mathcal{B}^C \text{ index } ps = \text{size } \{\# b \in \# \mathcal{B} . (\forall x \in ps . x \notin b) \# \}$

*<proof>*

**lemma** *complement-index-2:*

**assumes**  $\{x, y\} \subseteq \mathcal{V}$

**shows**  $\mathcal{B}^C \text{ index } \{x, y\} = \text{size } \{\# b \in \# \mathcal{B} . x \notin b \wedge y \notin b \# \}$

*<proof>*

**lemma** *complement-rep-number:*

**assumes**  $x \in \mathcal{V}$  **and**  $\mathcal{B} \text{ rep } x = r$

**shows**  $\mathcal{B}^C \text{ rep } x = b - r$

*<proof>*

**lemma** *complement-blocks-wf:*  $bl \in \# \mathcal{B}^C \implies bl \subseteq \mathcal{V}$

*<proof>*

**lemma** *complement-wf [intro]:* *incidence-system*  $\mathcal{V} \mathcal{B}^C$

*<proof>*

**interpretation** *sys-complement:* *incidence-system*  $\mathcal{V} \mathcal{B}^C$

*<proof>*

**end**

**context** *finite-incidence-system*

**begin**

**lemma** *block-complement-size:*  $b \subseteq \mathcal{V} \implies \text{card } (b^c) = \text{card } \mathcal{V} - \text{card } b$

*<proof>*

**lemma** *block-comp-incomplete:* *incomplete-block*  $bl \implies \text{card } (bl^c) > 0$

*<proof>*

**lemma** *block-comp-incomplete-nempty:* *incomplete-block*  $bl \implies bl^c \neq \{\}$

*<proof>*

**lemma** *incomplete-block-proper-subset:* *incomplete-block*  $bl \implies bl \subset \mathcal{V}$

*<proof>*

**lemma** *complement-finite:* *finite-incidence-system*  $\mathcal{V} \mathcal{B}^C$

*<proof>*

**interpretation** *comp-fin:* *finite-incidence-system*  $\mathcal{V} \mathcal{B}^C$

*<proof>*

**end**



**context** *design*  
**begin**  
**lemma** (in *design*) *complement-design*:  
 assumes  $\bigwedge bl . bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
 shows *design*  $\mathcal{V} (\mathcal{B}^C)$   
 <proof>  
**end**

## 2.7.2 Multiples

An easy way to construct new set systems is to simply multiply the block collection by some constant

**context** *incidence-system*  
**begin**

**abbreviation** *multiple-blocks* :: *nat*  $\Rightarrow$  'a set multiset **where**  
*multiple-blocks*  $n \equiv \text{repeat-mset } n \mathcal{B}$

**lemma** *multiple-block-in-original*:  $b \in \# \text{ multiple-blocks } n \implies b \in \# \mathcal{B}$   
 <proof>

**lemma** *multiple-block-in*:  $n > 0 \implies b \in \# \mathcal{B} \implies b \in \# \text{ multiple-blocks } n$   
 <proof>

**lemma** *multiple-blocks-gt*:  $n > 0 \implies \text{size } (\text{multiple-blocks } n) \geq \text{size } \mathcal{B}$   
 <proof>

**lemma** *block-original-count-le*:  $n > 0 \implies \text{count } \mathcal{B} \ b \leq \text{count } (\text{multiple-blocks } n) \ b$   
 <proof>

**lemma** *multiple-blocks-sub*:  $n > 0 \implies \mathcal{B} \subseteq \# (\text{multiple-blocks } n)$   
 <proof>

**lemma** *multiple-1-same*: *multiple-blocks* 1 =  $\mathcal{B}$   
 <proof>

**lemma** *multiple-unfold-1*: *multiple-blocks* (Suc  $n$ ) = (*multiple-blocks*  $n$ ) +  $\mathcal{B}$   
 <proof>

**lemma** *multiple-point-rep-num*: (*multiple-blocks*  $n$ ) *rep*  $x = (\mathcal{B} \text{ rep } x) * n$   
 <proof>

**lemma** *multiple-point-index*: (*multiple-blocks*  $n$ ) *index*  $ps = (\mathcal{B} \text{ index } ps) * n$   
 <proof>

**lemma** *repeat-mset-block-point-rel*:  $\bigwedge b \ x . b \in \# \text{ multiple-blocks } n \implies x \in b \implies x \in \mathcal{V}$   
 <proof>

**lemma** *multiple-is-wellformed: incidence-system*  $\mathcal{V}$  (*multiple-blocks*  $n$ )  
*<proof>*

**lemma** *multiple-blocks-num [simp]: size (multiple-blocks*  $n$ ) =  $n * b$   
*<proof>*

**interpretation** *mult-sys: incidence-system*  $\mathcal{V}$  (*multiple-blocks*  $n$ )  
*<proof>*

**lemma** *multiple-block-multiplicity [simp]: mult-sys.multiplicity*  $n$   $bl$  = (*multiplicity*  $bl$ ) \*  $n$   
*<proof>*

**lemma** *multiple-block-sizes-same:*  
  **assumes**  $n > 0$   
  **shows** *sys-block-sizes* = *mult-sys.sys-block-sizes*  $n$   
*<proof>*

**end**

**context** *finite-incidence-system*  
**begin**

**lemma** *multiple-is-finite: finite-incidence-system*  $\mathcal{V}$  (*multiple-blocks*  $n$ )  
*<proof>*

**end**

**context** *design*  
**begin**

**lemma** *multiple-is-design: design*  $\mathcal{V}$  (*multiple-blocks*  $n$ )  
*<proof>*

**end**

## 2.8 Simple Designs

Simple designs are those in which the multiplicity of each block is at most one. In other words, the block collection is a set. This can significantly ease reasoning.

**locale** *simple-incidence-system* = *incidence-system* +  
  **assumes** *simple [simp]:*  $bl \in \# \mathcal{B} \implies \text{multiplicity } bl = 1$

**begin**

**lemma** *simple-alt-def-all:*  $\forall bl \in \# \mathcal{B} . \text{multiplicity } bl = 1$   
*<proof>*

**lemma** *simple-blocks-eq-sup*:  $mset-set (design-support) = \mathcal{B}$   
*<proof>*

**lemma** *simple-block-size-eq-card*:  $b = card (design-support)$   
*<proof>*

**lemma** *points-index-simple-def*:  $\mathcal{B} \text{ index } ps = card \{b \in design-support . ps \subseteq b\}$   
*<proof>*

**lemma** *replication-num-simple-def*:  $\mathcal{B} \text{ rep } x = card \{b \in design-support . x \in b\}$   
*<proof>*

**end**

**locale** *simple-design* = *design* + *simple-incidence-system*

Additional reasoning about when something is not simple

**context** *incidence-system*

**begin**

**lemma** *simple-not-multiplicity*:  $b \in \# \mathcal{B} \implies multiplicity \ b > 1 \implies \neg \text{simple-incidence-system } \mathcal{V} \ \mathcal{B}$   
*<proof>*

**lemma** *multiple-not-simple*:  
  **assumes**  $n > 1$   
  **assumes**  $\mathcal{B} \neq \{\#\}$   
  **shows**  $\neg \text{simple-incidence-system } \mathcal{V} (\text{multiple-blocks } n)$   
*<proof>*

**end**

## 2.9 Proper Designs

Many types of designs rely on parameter conditions that only make sense for non-empty designs. i.e. designs with at least one block, and therefore given well-formed condition, at least one point. To this end we define the notion of a "proper" design

**locale** *proper-design* = *design* +  
  **assumes** *b-non-zero*:  $b \neq 0$   
**begin**

**lemma** *is-proper*: *proper-design*  $\mathcal{V} \ \mathcal{B}$  *<proof>*

**lemma** *v-non-zero*:  $v > 0$   
*<proof>*

**lemma** *b-positive*:  $b > 0$  *<proof>*

**lemma** *design-points-nempty*:  $\mathcal{V} \neq \{\}$   
*<proof>*

**lemma** *design-blocks-nempty*:  $\mathcal{B} \neq \{\#\}$   
*<proof>*

**end**

Intro rules for a proper design

**lemma** (**in** *design*) *proper-designI*[*intro*]:  $b \neq 0 \implies \text{proper-design } \mathcal{V} \mathcal{B}$   
*<proof>*

**lemma** *proper-designII*[*intro*]:  
**assumes** *design*  $\mathcal{V} \mathcal{B}$  **and**  $\mathcal{B} \neq \{\#\}$   
**shows** *proper-design*  $\mathcal{V} \mathcal{B}$   
*<proof>*

Reasoning on construction closure for proper designs

**context** *proper-design*  
**begin**

**lemma** *multiple-proper-design*:  
**assumes**  $n > 0$   
**shows** *proper-design*  $\mathcal{V}$  (*multiple-blocks*  $n$ )  
*<proof>*

**lemma** *complement-proper-design*:  
**assumes**  $\bigwedge bl . bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
**shows** *proper-design*  $\mathcal{V} \mathcal{B}^C$   
*<proof>*

**end**

**end**

**theory** *Design-Operations* **imports** *Design-Basics*  
**begin**

### 3 Design Operations

Incidence systems have a number of very typical computational operations which can be used for constructions in design theory. Definitions in this section are based off the handbook of combinatorial designs, hypergraph theory [2], and the GAP design theory library [5]

#### 3.1 Incidence system definitions

**context** *incidence-system*  
**begin**

The basic add point operation only affects the point set of a design

**definition** *add-point* :: 'a  $\Rightarrow$  'a set **where**  
*add-point* p  $\equiv$  insert p  $\mathcal{V}$

**lemma** *add-existing-point* [simp]: p  $\in$   $\mathcal{V} \implies$  *add-point* p =  $\mathcal{V}$   
 <proof>

**lemma** *add-point-wf*: incidence-system (*add-point* p)  $\mathcal{B}$   
 <proof>

An extension of the basic add point operation also adds the point to a given set of blocks

**definition** *add-point-to-blocks* :: 'a  $\Rightarrow$  'a set set  $\Rightarrow$  'a set multiset **where**  
*add-point-to-blocks* p bs  $\equiv$  {# (insert p b) | b  $\in$  #  $\mathcal{B}$  . b  $\in$  bs#} + {# b  $\in$  #  $\mathcal{B}$  . b  $\notin$  bs#}

**lemma** *add-point-blocks-blocks-alt*: *add-point-to-blocks* p bs =  
 image-mset (insert p) (filter-mset ( $\lambda$  b . b  $\in$  bs)  $\mathcal{B}$ ) + (filter-mset ( $\lambda$  b . b  $\notin$  bs)  $\mathcal{B}$ )  
 <proof>

**lemma** *add-point-existing-blocks*:  
**assumes** ( $\bigwedge$  bl . bl  $\in$  bs  $\implies$  p  $\in$  bl)  
**shows** *add-point-to-blocks* p bs =  $\mathcal{B}$   
 <proof>

**lemma** *add-new-point-rep-number*:  
**assumes** p  $\notin$   $\mathcal{V}$   
**shows** (*add-point-to-blocks* p bs) rep p = size {#b  $\in$  #  $\mathcal{B}$  . b  $\in$  bs#}  
 <proof>

**lemma** *add-point-blocks-wf*: incidence-system (*add-point* p) (*add-point-to-blocks* p bs)  
 <proof>

Basic (weak) delete point operation removes a point from both the point set and from any blocks that contain it to maintain wellformed property

**definition** *del-point* :: 'a  $\Rightarrow$  'a set **where**  
*del-point* p  $\equiv$   $\mathcal{V} - \{p\}$

**definition** *del-point-blocks*:: 'a  $\Rightarrow$  'a set multiset **where**  
*del-point-blocks* p  $\equiv$  {# (bl - {p}) . bl  $\in$  #  $\mathcal{B}$  #}

**lemma** *del-point-block-count*: size (*del-point-blocks* p) = size  $\mathcal{B}$   
 <proof>

**lemma** *remove-invalid-point-block*: p  $\notin$   $\mathcal{V} \implies$  bl  $\in$  #  $\mathcal{B} \implies$  bl - {p} = bl  
 <proof>

**lemma** *del-invalid-point*: p  $\notin$   $\mathcal{V} \implies$  (*del-point* p) =  $\mathcal{V}$

$\langle proof \rangle$

**lemma** *del-invalid-point-blocks*:  $p \notin \mathcal{V} \implies (\text{del-point-blocks } p) = \mathcal{B}$   
 $\langle proof \rangle$

**lemma** *delete-point-p-not-in-bl-blocks*:  $(\bigwedge bl. bl \in \# \mathcal{B} \implies p \notin bl) \implies (\text{del-point-blocks } p) = \mathcal{B}$   
 $\langle proof \rangle$

**lemma** *delete-point-blocks-wf*:  $b \in \# (\text{del-point-blocks } p) \implies b \subseteq \mathcal{V} - \{p\}$   
 $\langle proof \rangle$

**lemma** *delete-point-blocks-sub*:  
**assumes**  $b \in \# (\text{del-point-blocks } p)$   
**obtains**  $bl$  **where**  $bl \in \# \mathcal{B} \wedge b \subseteq bl$   
 $\langle proof \rangle$

**lemma** *delete-point-split-blocks*:  $\text{del-point-blocks } p =$   
 $\{\# bl \in \# \mathcal{B} . p \notin bl\# \} + \{\# bl - \{p\} \mid bl \in \# \mathcal{B} . p \in bl\# \}$   
 $\langle proof \rangle$

**lemma** *delete-point-index-eq*:  
**assumes**  $ps \subseteq (\text{del-point } p)$   
**shows**  $(\text{del-point-blocks } p) \text{ index } ps = \mathcal{B} \text{ index } ps$   
 $\langle proof \rangle$

**lemma** *delete-point-wf*: *incidence-system*  $(\text{del-point } p)$   $(\text{del-point-blocks } p)$   
 $\langle proof \rangle$

The concept of a strong delete point comes from hypergraph theory. When a point is deleted, any blocks containing it are also deleted

**definition** *str-del-point-blocks* ::  $'a \Rightarrow 'a$  set multiset **where**  
 $\text{str-del-point-blocks } p \equiv \{\# bl \in \# \mathcal{B} . p \notin bl\#\}$

**lemma** *str-del-point-blocks-alt*:  $\text{str-del-point-blocks } p = \mathcal{B} - \{\# bl \in \# \mathcal{B} . p \in bl\#\}$   
 $\langle proof \rangle$

**lemma** *delete-point-strong-block-in*:  $p \notin bl \implies bl \in \# \mathcal{B} \implies bl \in \# \text{str-del-point-blocks } p$   
 $\langle proof \rangle$

**lemma** *delete-point-strong-block-not-in*:  $p \in bl \implies bl \notin \# (\text{str-del-point-blocks } p)$   
 $\langle proof \rangle$

**lemma** *delete-point-strong-block-in-iff*:  $bl \in \# \mathcal{B} \implies bl \in \# \text{str-del-point-blocks } p$   
 $\iff p \notin bl$   
 $\langle proof \rangle$

**lemma** *delete-point-strong-block-subset*:  $str\text{-}del\text{-}point\text{-}blocks\ p \subseteq\# \mathcal{B}$   
 ⟨proof⟩

**lemma** *delete-point-strong-block-in-orig*:  $bl \in\# str\text{-}del\text{-}point\text{-}blocks\ p \implies bl \in\# \mathcal{B}$   
 ⟨proof⟩

**lemma** *delete-invalid-pt-strong-eq*:  $p \notin \mathcal{V} \implies \mathcal{B} = str\text{-}del\text{-}point\text{-}blocks\ p$   
 ⟨proof⟩

**lemma** *strong-del-point-index-alt*:  
 assumes  $ps \subseteq (del\text{-}point\ p)$   
 shows  $(str\text{-}del\text{-}point\text{-}blocks\ p)\ index\ ps =$   
 $\mathcal{B}\ index\ ps - \{\#\ bl \in\# \mathcal{B} . p \in bl\#\}\ index\ ps$   
 ⟨proof⟩

**lemma** *strong-del-point-incidence-wf*:  $incidence\text{-}system\ (del\text{-}point\ p)\ (str\text{-}del\text{-}point\text{-}blocks\ p)$   
 ⟨proof⟩

Add block operation

**definition** *add-block* ::  $'a\ set \Rightarrow 'a\ set\ multiset$  **where**  
 $add\text{-}block\ b \equiv \mathcal{B} + \{\#b\#\}$

**lemma** *add-block-alt*:  $add\text{-}block\ b = add\text{-}mset\ b\ \mathcal{B}$   
 ⟨proof⟩

**lemma** *add-block-rep-number-in*:  
 assumes  $x \in b$   
 shows  $(add\text{-}block\ b)\ rep\ x = \mathcal{B}\ rep\ x + 1$   
 ⟨proof⟩

**lemma** *add-block-rep-number-not-in*:  $x \notin b \implies (add\text{-}block\ b)\ rep\ x = \mathcal{B}\ rep\ x$   
 ⟨proof⟩

**lemma** *add-block-index-in*:  
 assumes  $ps \subseteq b$   
 shows  $(add\text{-}block\ b)\ index\ ps = \mathcal{B}\ index\ ps + 1$   
 ⟨proof⟩

**lemma** *add-block-index-not-in*:  $\neg (ps \subseteq b) \implies (add\text{-}block\ b)\ index\ ps = \mathcal{B}\ index\ ps$   
 ⟨proof⟩

Note the add block incidence system is defined slightly differently than textbook definitions due to the modification to the point set. This ensures the operation is closed, where otherwise a condition that  $b \subseteq \mathcal{V}$  would be required.

**lemma** *add-block-wf*:  $incidence\text{-}system\ (\mathcal{V} \cup b)\ (add\text{-}block\ b)$   
 ⟨proof⟩

**lemma** *add-block-wf-cond*:  $b \subseteq \mathcal{V} \implies \text{incidence-system } \mathcal{V} \text{ (add-block } b)$   
 ⟨proof⟩

Delete block removes a block from the block set. The point set is unchanged

**definition** *del-block* :: 'a set  $\Rightarrow$  'a set multiset **where**  
*del-block*  $b \equiv \mathcal{B} - \{\#b\# \}$

**lemma** *delete-block-subset*:  $(\text{del-block } b) \subseteq \# \mathcal{B}$   
 ⟨proof⟩

**lemma** *delete-invalid-block-eq*:  $b \notin \# \mathcal{B} \implies \text{del-block } b = \mathcal{B}$   
 ⟨proof⟩

**lemma** *delete-block-wf*: *incidence-system*  $\mathcal{V}$  (*del-block*  $b$ )  
 ⟨proof⟩

The strong delete block operation effectively deletes the block, as well as all points in that block

**definition** *str-del-block* :: 'a set  $\Rightarrow$  'a set multiset **where**  
*str-del-block*  $b \equiv \{\# \text{bl} - b \mid \text{bl} \in \# \mathcal{B} . \text{bl} \neq b \#\}$

**lemma** *strong-del-block-alt-def*: *str-del-block*  $b = \{\# \text{bl} - b . \text{bl} \in \# \text{removeAll-mset } b \mathcal{B} \#\}$   
 ⟨proof⟩

**lemma** *strong-del-block-wf*: *incidence-system*  $(\mathcal{V} - b)$  (*str-del-block*  $b$ )  
 ⟨proof⟩

**lemma** *str-del-block-del-point*:  
**assumes**  $\{x\} \notin \# \mathcal{B}$   
**shows** *str-del-block*  $\{x\} = (\text{del-point-blocks } x)$   
 ⟨proof⟩

## 3.2 Incidence System Interpretations

It is easy to interpret all operations as incidence systems in there own right. These can then be used to prove local properties on the new constructions, as well as reason on interactions between different operation sequences

**interpretation** *add-point-sys*: *incidence-system* *add-point*  $p \mathcal{B}$   
 ⟨proof⟩

**lemma** *add-point-sys-rep-numbers*: *add-point-sys.replication-numbers*  $p =$   
*replication-numbers*  $\cup \{\mathcal{B} \text{ rep } p\}$   
 ⟨proof⟩

**interpretation** *del-point-sys*: *incidence-system* *del-point*  $p \text{del-point-blocks } p$



*<proof>*

**interpretation** *add-block-sys: incidence-system*  $\mathcal{V} \cup bl$  *add-block*  $bl$   
*<proof>*

**interpretation** *del-block-sys: incidence-system*  $\mathcal{V}$  *del-block*  $bl$   
*<proof>*

**lemma** *add-del-block-inv:*  
**assumes**  $bl \subseteq \mathcal{V}$   
**shows** *add-block-sys.del-block*  $bl$   $bl = \mathcal{B}$   
*<proof>*

**lemma** *del-add-block-inv:*  $bl \in \# \mathcal{B} \implies$  *del-block-sys.add-block*  $bl$   $bl = \mathcal{B}$   
*<proof>*

**lemma** *del-invalid-add-block-eq:*  $bl \notin \# \mathcal{B} \implies$  *del-block-sys.add-block*  $bl$   $bl =$  *add-block*  $bl$   
*<proof>*

**lemma** *add-delete-point-inv:*  
**assumes**  $p \notin \mathcal{V}$   
**shows** *add-point-sys.del-point*  $p$   $p = \mathcal{V}$   
*<proof>*  
**end**

### 3.3 Operation Closure for Designs

**context** *finite-incidence-system*  
**begin**

**lemma** *add-point-finite: finite-incidence-system* (*add-point*  $p$ )  $\mathcal{B}$   
*<proof>*

**lemma** *add-point-to-blocks-finite: finite-incidence-system* (*add-point*  $p$ ) (*add-point-to-blocks*  $p$   $bs$ )  
*<proof>*

**lemma** *delete-point-finite:*  
*finite-incidence-system* (*del-point*  $p$ ) (*del-point-blocks*  $p$ )  
*<proof>*

**lemma** *del-point-order:*  
**assumes**  $p \in \mathcal{V}$   
**shows**  $card$  (*del-point*  $p$ )  $= v - 1$   
*<proof>*

**lemma** *strong-del-point-finite: finite-incidence-system* (*del-point*  $p$ ) (*str-del-point-blocks*  $p$ )

*<proof>*

**lemma** *add-block-fin: finite b  $\implies$  finite-incidence-system  $(\mathcal{V} \cup b)$  (add-block b)*  
*<proof>*

**lemma** *add-block-fin-cond:  $b \subseteq \mathcal{V} \implies$  finite-incidence-system  $\mathcal{V}$  (add-block b)*  
*<proof>*

**lemma** *delete-block-fin-incidence-sys: finite-incidence-system  $\mathcal{V}$  (del-block b)*  
*<proof>*

**lemma** *strong-del-block-fin: finite-incidence-system  $(\mathcal{V} - b)$  (str-del-block b)*  
*<proof>*

**end**

**context** *design*  
**begin**

**lemma** *add-point-design: design (add-point p)  $\mathcal{B}$*   
*<proof>*

**lemma** *delete-point-design:*  
**assumes**  $(\bigwedge bl . bl \in \# \mathcal{B} \implies p \in bl \implies \text{card } bl \geq 2)$   
**shows** *design (del-point p) (del-point-blocks p)*  
*<proof>*

**lemma** *strong-del-point-design: design (del-point p) (str-del-point-blocks p)*  
*<proof>*

**lemma** *add-block-design:*  
**assumes** *finite bl*  
**assumes**  $bl \neq \{\}$   
**shows** *design  $(\mathcal{V} \cup bl)$  (add-block bl)*  
*<proof>*

**lemma** *add-block-design-cond:*  
**assumes**  $bl \subseteq \mathcal{V}$  **and**  $bl \neq \{\}$   
**shows** *design  $\mathcal{V}$  (add-block bl)*  
*<proof>*

**lemma** *delete-block-design: design  $\mathcal{V}$  (del-block bl)*  
*<proof>*

**lemma** *strong-del-block-des:*  
**assumes**  $\bigwedge bl . bl \in \# \mathcal{B} \implies \neg (bl \subset b)$   
**shows** *design  $(\mathcal{V} - b)$  (str-del-block b)*  
*<proof>*

**end**

```

context proper-design
begin
lemma delete-point-proper:
  assumes  $\bigwedge bl. bl \in \# \mathcal{B} \implies p \in bl \implies 2 \leq \text{card } bl$ 
  shows proper-design (del-point  $p$ ) (del-point-blocks  $p$ )
   $\langle \text{proof} \rangle$ 

lemma strong-delete-point-proper:
  assumes  $\bigwedge bl. bl \in \# \mathcal{B} \implies p \in bl \implies 2 \leq \text{card } bl$ 
  assumes  $\mathcal{B} \text{ rep } p < b$ 
  shows proper-design (del-point  $p$ ) (str-del-point-blocks  $p$ )
   $\langle \text{proof} \rangle$ 

end

```

### 3.4 Combining Set Systems

Similar to multiple, another way to construct a new set system is to combine two existing ones. We introduce a new locale enabling us to reason on two different incidence systems

```

locale two-set-systems = sys1: incidence-system  $\mathcal{V}$   $\mathcal{B}$  + sys2: incidence-system  $\mathcal{V}'$   $\mathcal{B}'$ 

```

```

  for  $\mathcal{V} :: ('a \text{ set})$  and  $\mathcal{B}$  and  $\mathcal{V}' :: ('a \text{ set})$  and  $\mathcal{B}'$ 

```

```

begin

```

```

abbreviation combine-points  $\equiv \mathcal{V} \cup \mathcal{V}'$ 

```

```

notation combine-points ( $\mathcal{V}^+$ )

```

```

abbreviation combine-blocks  $\equiv \mathcal{B} + \mathcal{B}'$ 

```

```

notation combine-blocks ( $\mathcal{B}^+$ )

```

```

sublocale combine-sys: incidence-system  $\mathcal{V}^+$   $\mathcal{B}^+$ 

```

```

   $\langle \text{proof} \rangle$ 

```

```

lemma combine-points-index:  $\mathcal{B}^+ \text{ index } ps = \mathcal{B} \text{ index } ps + \mathcal{B}' \text{ index } ps$ 

```

```

   $\langle \text{proof} \rangle$ 

```

```

lemma combine-rep-number:  $\mathcal{B}^+ \text{ rep } x = \mathcal{B} \text{ rep } x + \mathcal{B}' \text{ rep } x$ 

```

```

   $\langle \text{proof} \rangle$ 

```

```

lemma combine-multiple1:  $\mathcal{V} = \mathcal{V}' \implies \mathcal{B} = \mathcal{B}' \implies \mathcal{B}^+ = \text{sys1.multiple-blocks } 2$ 

```

```

   $\langle \text{proof} \rangle$ 

```

```

lemma combine-multiple2:  $\mathcal{V} = \mathcal{V}' \implies \mathcal{B} = \mathcal{B}' \implies \mathcal{B}^+ = \text{sys2.multiple-blocks } 2$ 

```

```

   $\langle \text{proof} \rangle$ 

```

**lemma** *combine-multiplicity*:  $\text{combine-sys.multiplicity } b = \text{sys1.multiplicity } b + \text{sys2.multiplicity } b$   
 ⟨proof⟩

**lemma** *combine-block-sizes*:  $\text{combine-sys.sys-block-sizes} = \text{sys1.sys-block-sizes} \cup \text{sys2.sys-block-sizes}$   
 ⟨proof⟩

**end**

**locale** *two-fin-set-systems* = *two-set-systems*  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}'$  + *sys1: finite-incidence-system*  $\mathcal{V} \mathcal{B}$  +  
*sys2: finite-incidence-system*  $\mathcal{V}' \mathcal{B}'$  **for**  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}'$   
**begin**

**sublocale** *combine-fin-sys*: *finite-incidence-system*  $\mathcal{V}^+ \mathcal{B}^+$   
 ⟨proof⟩

**lemma** *combine-order*:  $\text{card } (\mathcal{V}^+) \geq \text{card } \mathcal{V}$   
 ⟨proof⟩

**lemma** *combine-order-2*:  $\text{card } (\mathcal{V}^+) \geq \text{card } \mathcal{V}'$   
 ⟨proof⟩

**end**

**locale** *two-designs* = *two-fin-set-systems*  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}'$  + *des1: design*  $\mathcal{V} \mathcal{B}$  +  
*des2: design*  $\mathcal{V}' \mathcal{B}'$  **for**  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}'$   
**begin**

**sublocale** *combine-des*: *design*  $\mathcal{V}^+ \mathcal{B}^+$   
 ⟨proof⟩

**end**

**locale** *two-designs-proper* = *two-designs* +  
**assumes** *blocks-nempty*:  $\mathcal{B} \neq \{\#\} \vee \mathcal{B}' \neq \{\#\}$   
**begin**

**lemma** *des1-is-proper*:  $\mathcal{B} \neq \{\#\} \implies \text{proper-design } \mathcal{V} \mathcal{B}$   
 ⟨proof⟩

**lemma** *des2-is-proper*:  $\mathcal{B}' \neq \{\#\} \implies \text{proper-design } \mathcal{V}' \mathcal{B}'$   
 ⟨proof⟩

**lemma** *min-one-proper-design*:  $\text{proper-design } \mathcal{V} \mathcal{B} \vee \text{proper-design } \mathcal{V}' \mathcal{B}'$   
 ⟨proof⟩

```

sublocale combine-proper-des: proper-design  $\mathcal{V}^+ \mathcal{B}^+$ 
   $\langle$ proof $\rangle$ 
end

end

```

## 4 Block and Balanced Designs

We define a selection of the many different types of block and balanced designs, building up to properties required for defining a BIBD, in addition to several base generalisations

```

theory Block-Designs imports Design-Operations
begin

```

### 4.1 Block Designs

A block design is a design where all blocks have the same size.

#### 4.1.1 K Block Designs

An important generalisation of a typical block design is the  $\mathcal{K}$  block design, where all blocks must have a size  $x$  where  $x \in \mathcal{K}$

```

locale K-block-design = proper-design +
  fixes sizes :: nat set ( $\mathcal{K}$ )
  assumes block-sizes:  $bl \in \# \mathcal{B} \implies \text{card } bl \in \mathcal{K}$ 
  assumes positive-ints:  $x \in \mathcal{K} \implies x > 0$ 
begin

```

```

lemma sys-block-size-subset:  $\text{sys-block-sizes} \subseteq \mathcal{K}$ 
   $\langle$ proof $\rangle$ 

```

```

end

```

#### 4.1.2 Uniform Block Design

The typical uniform block design is defined below

```

locale block-design = proper-design +
  fixes u-block-size :: nat ( $k$ )
  assumes uniform [simp]:  $bl \in \# \mathcal{B} \implies \text{card } bl = k$ 
begin

```

```

lemma k-non-zero:  $k \geq 1$ 
   $\langle$ proof $\rangle$ 

```

```

lemma uniform-alt-def-all:  $\forall bl \in \# \mathcal{B} . \text{card } bl = k$ 
   $\langle$ proof $\rangle$ 

```

**lemma** *uniform-unfold-point-set*:  $bl \in \# \mathcal{B} \implies \text{card } \{p \in \mathcal{V}. p \in bl\} = k$   
 ⟨proof⟩

**lemma** *uniform-unfold-point-set-mset*:  $bl \in \# \mathcal{B} \implies \text{size } \{\#p \in \# \text{mset-set } \mathcal{V}. p \in bl \#\} = k$   
 ⟨proof⟩

**lemma** *sys-block-sizes-uniform* [simp]:  $\text{sys-block-sizes} = \{k\}$   
 ⟨proof⟩

**lemma** *sys-block-sizes-uniform-single*: *is-singleton* (*sys-block-sizes*)  
 ⟨proof⟩

**lemma** *uniform-size-incomp*:  $k \leq v - 1 \implies bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
 ⟨proof⟩

**lemma** *uniform-complement-block-size*:  
 assumes  $bl \in \# \mathcal{B}^C$   
 shows  $\text{card } bl = v - k$   
 ⟨proof⟩

**lemma** *uniform-complement*[intro]:  
 assumes  $k \leq v - 1$   
 shows *block-design*  $\mathcal{V} \mathcal{B}^C (v - k)$   
 ⟨proof⟩

**lemma** *block-size-lt-v*:  $k \leq v$   
 ⟨proof⟩

**end**

**lemma** (in *proper-design*) *block-designI*[intro]:  $(\bigwedge bl . bl \in \# \mathcal{B} \implies \text{card } bl = k) \implies \text{block-design } \mathcal{V} \mathcal{B} k$   
 ⟨proof⟩

**context** *block-design*  
**begin**

**lemma** *block-design-multiple*:  $n > 0 \implies \text{block-design } \mathcal{V} (\text{multiple-blocks } n) k$   
 ⟨proof⟩

**end**

A uniform block design is clearly a type of *K\_block\_design* with a singleton *K* set

**sublocale** *block-design*  $\subseteq$  *K-block-design*  $\mathcal{V} \mathcal{B} \{k\}$   
 ⟨proof⟩

### 4.1.3 Incomplete Designs

An incomplete design is a design where  $k < v$ , i.e. no block is equal to the point set

**locale** *incomplete-design* = *block-design* +  
**assumes** *incomplete*:  $k < v$

**begin**

**lemma** *incomplete-imp-incomp-block*:  $bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
 $\langle \text{proof} \rangle$

**lemma** *incomplete-imp-proper-subset*:  $bl \in \# \mathcal{B} \implies bl \subset \mathcal{V}$   
 $\langle \text{proof} \rangle$

**end**

**lemma** (in *block-design*) *incomplete-designI*[*intro*]:  $k < v \implies \text{incomplete-design } \mathcal{V} \mathcal{B} k$   
 $\langle \text{proof} \rangle$

**context** *incomplete-design*

**begin**

**lemma** *multiple-incomplete*:  $n > 0 \implies \text{incomplete-design } \mathcal{V} (\text{multiple-blocks } n) k$   
 $\langle \text{proof} \rangle$

**lemma** *complement-incomplete*:  $\text{incomplete-design } \mathcal{V} (\mathcal{B}^C) (v - k)$   
 $\langle \text{proof} \rangle$

**end**

## 4.2 Balanced Designs

$t$ -wise balance is a design with the property that all point subsets of size  $t$  occur in  $\lambda_t$  blocks

**locale** *t-wise-balance* = *proper-design* +  
**fixes** *grouping* ::  $\text{nat } (t)$  **and** *index* ::  $\text{nat } (\Lambda_t)$   
**assumes** *t-non-zero*:  $t \geq 1$   
**assumes** *t-lt-order*:  $t \leq v$   
**assumes** *balanced* [*simp*]:  $ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps = \Lambda_t$

**begin**

**lemma** *t-non-zero-suc*:  $t \geq \text{Suc } 0$   
 $\langle \text{proof} \rangle$

**lemma** *balanced-alt-def-all*:  $\forall ps \subseteq \mathcal{V} . \text{card } ps = t \longrightarrow \mathcal{B} \text{ index } ps = \Lambda_t$   
 $\langle \text{proof} \rangle$

**end**

**lemma** (in *proper-design*) *t-wise-balanceI*[*intro*]:  $t \leq v \implies t \geq 1 \implies$   
 $(\bigwedge ps . ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps = \Lambda_t) \implies t\text{-wise-balance } \mathcal{V} \mathcal{B} t$   
 $\Lambda_t$   
 ⟨*proof*⟩

**context** *t-wise-balance*  
**begin**

**lemma** *obtain-t-subset-points*:  
**obtains** *T* **where**  $T \subseteq \mathcal{V}$   $\text{card } T = t$  *finite T*  
 ⟨*proof*⟩

**lemma** *multiple-t-wise-balance-index* [*simp*]:  
**assumes**  $ps \subseteq \mathcal{V}$   
**assumes**  $\text{card } ps = t$   
**shows** (*multiple-blocks n*)  $\text{index } ps = \Lambda_t * n$   
 ⟨*proof*⟩

**lemma** *multiple-t-wise-balance*:  
**assumes**  $n > 0$   
**shows** *t-wise-balance*  $\mathcal{V}$  (*multiple-blocks n*)  $t$   $(\Lambda_t * n)$   
 ⟨*proof*⟩

**lemma** *twice-set-pair-index*:  $ps \subseteq \mathcal{V} \implies ps2 \subseteq \mathcal{V} \implies ps \neq ps2 \implies \text{card } ps = t$   
 $\implies \text{card } ps2 = t$   
 $\implies \mathcal{B} \text{ index } ps = \mathcal{B} \text{ index } ps2$   
 ⟨*proof*⟩

**lemma** *t-wise-balance-alt*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps = l2$   
 $\implies (\bigwedge ps . ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps = l2)$   
 ⟨*proof*⟩

**lemma** *index-1-imp-mult-1* [*simp*]:  
**assumes**  $\Lambda_t = 1$   
**assumes**  $bl \in \# \mathcal{B}$   
**assumes**  $\text{card } bl \geq t$   
**shows** *multiplicity*  $bl = 1$   
 ⟨*proof*⟩

**end**

#### 4.2.1 Sub-types of t-wise balance

Pairwise balance is when  $t = 2$ . These are commonly of interest

**locale** *pairwise-balance* = *t-wise-balance*  $\mathcal{V} \mathcal{B} 2 \Lambda$   
**for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ ) **and** *index* ( $\Lambda$ )

We can combine the balance properties with *K\_block* design to define



tBD's (t-wise balanced designs), and PBD's (pairwise balanced designs)

**locale**  $tBD = t\text{-wise-balance} + K\text{-block-design} +$   
**assumes**  $block\text{-size-gt-t}: k \in \mathcal{K} \implies k \geq t$

**locale**  $\Lambda\text{-PBD} = pairwise\text{-balance} + K\text{-block-design} +$   
**assumes**  $block\text{-size-gt-t}: k \in \mathcal{K} \implies k \geq 2$

**sublocale**  $\Lambda\text{-PBD} \subseteq tBD \vee \mathcal{B} \geq \Lambda \mathcal{K}$   
*<proof>*

**locale**  $PBD = \Lambda\text{-PBD} \vee \mathcal{B} \geq 1 \mathcal{K}$  **for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ ) **and**  
*sizes* ( $\mathcal{K}$ )

**begin**

**lemma** *multiplicity-is-1:*

**assumes**  $bl \in \# \mathcal{B}$

**shows** *multiplicity*  $bl = 1$

*<proof>*

**end**

**sublocale**  $PBD \subseteq simple\text{-design}$   
*<proof>*

PBD's are often only used in the case where  $k$  is uniform, defined here.

**locale**  $k\text{-}\Lambda\text{-PBD} = pairwise\text{-balance} + block\text{-design} +$   
**assumes**  $block\text{-size-t}: 2 \leq k$

**sublocale**  $k\text{-}\Lambda\text{-PBD} \subseteq \Lambda\text{-PBD} \vee \mathcal{B} \geq \{k\}$   
*<proof>*

**locale**  $k\text{-PBD} = k\text{-}\Lambda\text{-PBD} \vee \mathcal{B} \geq 1 k$  **for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ ) **and**  
*u-block-size* ( $k$ )

**sublocale**  $k\text{-PBD} \subseteq PBD \vee \mathcal{B} \geq \{k\}$   
*<proof>*

## 4.2.2 Covering and Packing Designs

Covering and packing designs involve a looser balance restriction. Upper/lower bounds are placed on the points index, instead of a strict equality

A t-covering design is a relaxed version of a tBD, where, for all point subsets of size t, a lower bound is put on the points index

**locale**  $t\text{-covering-design} = block\text{-design} +$   
**fixes**  $grouping :: nat (t)$   
**fixes**  $min\text{-index} :: nat (\Lambda_t)$   
**assumes**  $covering: ps \subseteq \mathcal{V} \implies card\ ps = t \implies \mathcal{B}\ index\ ps \geq \Lambda_t$   
**assumes**  $block\text{-size-t}: t \leq k$   
**assumes**  $t\text{-non-zero}: t \geq 1$

**begin**

**lemma** *covering-alt-def-all*:  $\forall ps \subseteq \mathcal{V} . \text{card } ps = t \longrightarrow \mathcal{B} \text{ index } ps \geq \Lambda_t$   
*<proof>*

**end**

**lemma** (**in** *block-design*) *t-covering-designI* [*intro*]:  $t \leq k \implies t \geq 1 \implies$   
 $(\bigwedge ps . ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps \geq \Lambda_t) \implies t\text{-covering-design } \mathcal{V} \mathcal{B}$   
 $k \ t \ \Lambda_t$   
*<proof>*

A  $t$ -packing design is a relaxed version of a  $t$ BD, where, for all point subsets of size  $t$ , an upper bound is put on the points index

**locale** *t-packing-design* = *block-design* +  
**fixes** *grouping* :: *nat* ( $t$ )  
**fixes** *min-index* :: *nat* ( $\Lambda_t$ )  
**assumes** *packing*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps \leq \Lambda_t$   
**assumes** *block-size-t*:  $t \leq k$   
**assumes** *t-non-zero*:  $t \geq 1$   
**begin**

**lemma** *packing-alt-def-all*:  $\forall ps \subseteq \mathcal{V} . \text{card } ps = t \longrightarrow \mathcal{B} \text{ index } ps \leq \Lambda_t$   
*<proof>*

**end**

**lemma** (**in** *block-design*) *t-packing-designI* [*intro*]:  $t \leq k \implies t \geq 1 \implies$   
 $(\bigwedge ps . ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B} \text{ index } ps \leq \Lambda_t) \implies t\text{-packing-design } \mathcal{V} \mathcal{B}$   
 $k \ t \ \Lambda_t$   
*<proof>*

**lemma** *packing-covering-imp-balance*:  
**assumes** *t-packing-design*  $V \ B \ k \ t \ \Lambda_t$   
**assumes** *t-covering-design*  $V \ B \ k \ t \ \Lambda_t$   
**shows** *t-wise-balance*  $V \ B \ t \ \Lambda_t$   
*<proof>*

### 4.3 Constant Replication Design

When the replication number for all points in a design is constant, it is the design replication number.

**locale** *constant-rep-design* = *proper-design* +  
**fixes** *design-rep-number* :: *nat* ( $r$ )  
**assumes** *rep-number* [*simp*]:  $x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x = r$

**begin**

**lemma** *rep-number-alt-def-all*:  $\forall x \in \mathcal{V} . \mathcal{B} \text{ rep } x = r$

*<proof>*

**lemma** *rep-number-unfold-set*:  $x \in \mathcal{V} \implies \text{size } \{\#bl \in \# \mathcal{B} . x \in bl\# \} = r$   
*<proof>*

**lemma** *rep-numbers-constant* [*simp*]: *replication-numbers* = {r}  
*<proof>*

**lemma** *replication-number-single*: *is-singleton* (*replication-numbers*)  
*<proof>*

**lemma** *constant-rep-point-pair*:  $x1 \in \mathcal{V} \implies x2 \in \mathcal{V} \implies x1 \neq x2 \implies \mathcal{B} \text{ rep } x1 = \mathcal{B} \text{ rep } x2$   
*<proof>*

**lemma** *constant-rep-alt*:  $x1 \in \mathcal{V} \implies \mathcal{B} \text{ rep } x1 = r2 \implies (\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x = r2)$   
*<proof>*

**lemma** *constant-rep-point-not-0*:  
  **assumes**  $x \in \mathcal{V}$   
  **shows**  $\mathcal{B} \text{ rep } x \neq 0$   
*<proof>*

**lemma** *rep-not-zero*:  $r \neq 0$   
*<proof>*

**lemma** *r-gzero*:  $r > 0$   
*<proof>*

**lemma** *r-lt-eq-b*:  $r \leq b$   
*<proof>*

**lemma** *complement-rep-number*:  
  **assumes**  $\bigwedge bl . bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$   
  **shows** *constant-rep-design*  $\mathcal{V} \mathcal{B}^C (b - r)$   
*<proof>*

**lemma** *multiple-rep-number*:  
  **assumes**  $n > 0$   
  **shows** *constant-rep-design*  $\mathcal{V} (\text{multiple-blocks } n) (r * n)$   
*<proof>*  
**end**

**lemma** (**in** *proper-design*) *constant-rep-designI* [*intro*]:  $(\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x = r) \implies \text{constant-rep-design } \mathcal{V} \mathcal{B} r$   
*<proof>*

## 4.4 T-designs

All the before mentioned designs build up to the concept of a  $t$ -design, which has uniform block size and is  $t$ -wise balanced. We limit  $t$  to be less than  $k$ , so the balance condition has relevance

**locale**  $t$ -design = incomplete-design +  $t$ -wise-balance +  
**assumes**  $block\text{-}size\text{-}t$ :  $t \leq k$   
**begin**

**lemma**  $point\text{-}indices\text{-}balanced$ :  $point\text{-}indices\ t = \{\Lambda_t\}$   
 $\langle proof \rangle$

**lemma**  $point\text{-}indices\text{-}singleton$ :  $is\text{-}singleton\ (point\text{-}indices\ t)$   
 $\langle proof \rangle$

**end**

**lemma**  $t$ -designI [intro]:  
**assumes**  $incomplete\text{-}design\ V\ B\ k$   
**assumes**  $t\text{-}wise\text{-}balance\ V\ B\ t\ \Lambda_t$   
**assumes**  $t \leq k$   
**shows**  $t\text{-}design\ V\ B\ k\ t\ \Lambda_t$   
 $\langle proof \rangle$

**sublocale**  $t$ -design  $\subseteq$   $t$ -covering-design  $\mathcal{V}\ \mathcal{B}\ k\ t\ \Lambda_t$   
 $\langle proof \rangle$

**sublocale**  $t$ -design  $\subseteq$   $t$ -packing-design  $\mathcal{V}\ \mathcal{B}\ k\ t\ \Lambda_t$   
 $\langle proof \rangle$

**lemma**  $t$ -design-pack-cov [intro]:  
**assumes**  $k < card\ V$   
**assumes**  $t\text{-}covering\text{-}design\ V\ B\ k\ t\ \Lambda_t$   
**assumes**  $t\text{-}packing\text{-}design\ V\ B\ k\ t\ \Lambda_t$   
**shows**  $t\text{-}design\ V\ B\ k\ t\ \Lambda_t$   
 $\langle proof \rangle$

**sublocale**  $t$ -design  $\subseteq$   $tBD\ \mathcal{V}\ \mathcal{B}\ t\ \Lambda_t\ \{k\}$   
 $\langle proof \rangle$

**context**  $t$ -design  
**begin**

**lemma**  $multiple\text{-}t\text{-}design$ :  $n > 0 \implies t\text{-}design\ \mathcal{V}\ (multiple\text{-}blocks\ n)\ k\ t\ (\Lambda_t * n)$   
 $\langle proof \rangle$

**lemma**  $t$ -design-min-v:  $v > 1$   
 $\langle proof \rangle$

**end**

## 4.5 Steiner Systems

Steiner systems are a special type of  $t$ -design where  $\Lambda_t = 1$

**locale** *steiner-system* = *t-design*  $\mathcal{V} \mathcal{B} k t 1$   
for *point-set* ( $\mathcal{V}$ ) and *block-collection* ( $\mathcal{B}$ ) and *u-block-size* ( $k$ ) and *grouping* ( $t$ )

**begin**

**lemma** *block-multiplicity* [*simp*]:  
assumes  $bl \in \# \mathcal{B}$   
shows *multiplicity*  $bl = 1$   
(*proof*)

**end**

**sublocale** *steiner-system*  $\subseteq$  *simple-design*  
(*proof*)

**lemma** (*in t-design*) *steiner-systemI*[*intro*]:  $\Lambda_t = 1 \implies$  *steiner-system*  $\mathcal{V} \mathcal{B} k t$   
(*proof*)

## 4.6 Combining block designs

We define some closure properties for various block designs under the combine operator. This is done using locales to reason on multiple instances of the same type of design, building on what was presented in the design operations theory

**locale** *two-t-wise-eq-points* = *two-designs-proper*  $\mathcal{V} \mathcal{B} \mathcal{V} \mathcal{B}' +$  *des1: t-wise-balance*  
 $\mathcal{V} \mathcal{B} t \Lambda_t +$   
*des2: t-wise-balance*  $\mathcal{V} \mathcal{B}' t \Lambda_t'$  for  $\mathcal{V} \mathcal{B} t \Lambda_t \mathcal{B}' \Lambda_t'$

**begin**

**lemma** *combine-t-wise-balance-index*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = t \implies \mathcal{B}^+$  *index*  $ps =$   
 $(\Lambda_t + \Lambda_t')$   
(*proof*)

**lemma** *combine-t-wise-balance*: *t-wise-balance*  $\mathcal{V}^+ \mathcal{B}^+ t (\Lambda_t + \Lambda_t')$   
(*proof*)

**sublocale** *combine-t-wise-des*: *t-wise-balance*  $\mathcal{V}^+ \mathcal{B}^+ t (\Lambda_t + \Lambda_t')$   
(*proof*)

**end**

**locale** *two-k-block-designs* = *two-designs-proper*  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}' +$  *des1: block-design*  $\mathcal{V}$   
 $\mathcal{B} k +$

*des2: block-design  $\mathcal{V}' \mathcal{B}' k$  for  $\mathcal{V} \mathcal{B} k \mathcal{V}' \mathcal{B}'$*   
**begin**

**lemma** *block-design-combine: block-design  $\mathcal{V}^+ \mathcal{B}^+ k$*   
 *$\langle proof \rangle$*

**sublocale** *combine-block-des: block-design  $\mathcal{V}^+ \mathcal{B}^+ k$*   
 *$\langle proof \rangle$*

**end**

**locale** *two-rep-designs-eq-points = two-designs-proper  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}' + des1: constant-rep-design \mathcal{V} \mathcal{B} r +$*   
 *$des2: constant-rep-design \mathcal{V} \mathcal{B}' r'$  for  $\mathcal{V} \mathcal{B} r \mathcal{B}' r'$*   
**begin**

**lemma** *combine-rep-number: constant-rep-design  $\mathcal{V}^+ \mathcal{B}^+ (r + r')$*   
 *$\langle proof \rangle$*

**sublocale** *combine-const-rep: constant-rep-design  $\mathcal{V}^+ \mathcal{B}^+ (r + r')$*   
 *$\langle proof \rangle$*

**end**

**locale** *two-incomplete-designs = two-k-block-designs  $\mathcal{V} \mathcal{B} k \mathcal{V}' \mathcal{B}' + des1: incomplete-design \mathcal{V} \mathcal{B} k +$*   
 *$des2: incomplete-design \mathcal{V}' \mathcal{B}' k$  for  $\mathcal{V} \mathcal{B} k \mathcal{V}' \mathcal{B}'$*   
**begin**

**lemma** *combine-is-incomplete: incomplete-design  $\mathcal{V}^+ \mathcal{B}^+ k$*   
 *$\langle proof \rangle$*

**sublocale** *combine-incomplete: incomplete-design  $\mathcal{V}^+ \mathcal{B}^+ k$*   
 *$\langle proof \rangle$*

**end**

**locale** *two-t-designs-eq-points = two-incomplete-designs  $\mathcal{V} \mathcal{B} k \mathcal{V}' \mathcal{B}'$*   
 *$+ two-t-wise-eq-points \mathcal{V} \mathcal{B} t \Lambda_t \mathcal{B}' \Lambda_t' + des1: t-design \mathcal{V} \mathcal{B} k t \Lambda_t +$*   
 *$des2: t-design \mathcal{V} \mathcal{B}' k t \Lambda_t'$  for  $\mathcal{V} \mathcal{B} k \mathcal{B}' t \Lambda_t \Lambda_t'$*   
**begin**

**lemma** *combine-is-t-des: t-design  $\mathcal{V}^+ \mathcal{B}^+ k t (\Lambda_t + \Lambda_t')$*   
 *$\langle proof \rangle$*

**sublocale** *combine-t-des: t-design  $\mathcal{V}^+ \mathcal{B}^+ k t (\Lambda_t + \Lambda_t')$*   
 *$\langle proof \rangle$*

**end**

**end**

**theory** *BIBD* **imports** *Block-Designs*  
**begin**

## 5 BIBD's

BIBD's are perhaps the most commonly studied type of design in combinatorial design theory, and usually the first type of design explored in a design theory course. These designs are a type of  $t$ -design, where  $t = 2$

### 5.1 BIBD Basics

**locale** *bibd = t-design*  $\mathcal{V} \ \mathcal{B} \ k \ 2 \ \Lambda$   
**for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ )  
**and** *u-block-size* ( $k$ ) **and** *index* ( $\Lambda$ )

**begin**

**lemma** *min-block-size-2*:  $k \geq 2$   
 $\langle$ *proof* $\rangle$

**lemma** *points-index-pair*:  $y \in \mathcal{V} \implies x \in \mathcal{V} \implies x \neq y \implies \text{size} (\{\# \ bl \in \# \ \mathcal{B} . \{x, y\} \subseteq bl\#\}) = \Lambda$   
 $\langle$ *proof* $\rangle$

**lemma** *index-one-empty-rm-bl* [*simp*]:  
**assumes**  $\Lambda = 1$  **and**  $bl \in \# \ \mathcal{B}$  **and**  $p \subseteq bl$  **and**  $\text{card } p = 2$   
**shows**  $\{\# \ bl \in \# \ \text{remove1-mset } bl \ \mathcal{B} . p \subseteq bl\#\} = \{\#\}$   
 $\langle$ *proof* $\rangle$

**lemma** *index-one-alt-bl-not-exist*:  
**assumes**  $\Lambda = 1$  **and**  $bl \in \# \ \mathcal{B}$  **and**  $p \subseteq bl$  **and**  $\text{card } p = 2$   
**shows**  $\bigwedge \ bl. \ bl \in \# \ \text{remove1-mset } bl \ \mathcal{B} \implies \neg (p \subseteq bl)$   
 $\langle$ *proof* $\rangle$

### 5.2 Necessary Conditions for Existence

The necessary conditions on the existence of a  $(v, k, \lambda)$ -bibd are one of the fundamental first theorems on designs. Proofs based off MATH3301 lecture notes [4] and Stinson [6]

**lemma** *necess-cond-1-rhs*:  
**assumes**  $x \in \mathcal{V}$   
**shows**  $\text{size} (\{\# \ p \in \# \ (\text{mset-set } (\mathcal{V} - \{x\}) \times \# \ \{\# \ bl \in \# \ \mathcal{B} . x \in bl \ \#\}). \text{fst } p \in \text{snd } p\#\}) = \Lambda * (v - 1)$   
 $\langle$ *proof* $\rangle$

**lemma** *necess-cond-1-lhs*:

**assumes**  $x \in \mathcal{V}$

**shows**  $\text{size} (\{\# p \in \# (\text{mset-set } (\mathcal{V} - \{x\}) \times \# \{\# bl \in \# \mathcal{B} . x \in bl \#\}) . \text{fst } p \in \text{snd } p\#\})$

$= (\mathcal{B} \text{ rep } x) * (k - 1)$

$(\text{is size } (\{\# p \in \# (?M \times \# ?B) . \text{fst } p \in \text{snd } p\#\}) = (\mathcal{B} \text{ rep } x) * (k - 1) )$

$\langle \text{proof} \rangle$

**lemma** *r-constant*:  $x \in \mathcal{V} \implies (\mathcal{B} \text{ rep } x) * (k - 1) = \Lambda * (v - 1)$

$\langle \text{proof} \rangle$

**lemma** *replication-number-value*:

**assumes**  $x \in \mathcal{V}$

**shows**  $(\mathcal{B} \text{ rep } x) = \Lambda * (v - 1) \text{ div } (k - 1)$

$\langle \text{proof} \rangle$

**lemma** *r-constant-alt*:  $\forall x \in \mathcal{V} . \mathcal{B} \text{ rep } x = \Lambda * (v - 1) \text{ div } (k - 1)$

$\langle \text{proof} \rangle$

**end**

Using the first necessary condition, it is possible to show that a bibd has a constant replication number

**sublocale**  $\text{bibd} \subseteq \text{constant-rep-design } \mathcal{V} \mathcal{B} (\Lambda * (v - 1) \text{ div } (k - 1))$

$\langle \text{proof} \rangle$

**lemma** *(in t-design) bibdI [intro]*:  $t = 2 \implies \text{bibd } \mathcal{V} \mathcal{B} k \Lambda_t$

$\langle \text{proof} \rangle$

**context** *bibd*

**begin**

**abbreviation**  $r \equiv (\Lambda * (v - 1) \text{ div } (k - 1))$

**lemma** *necessary-condition-one*:

**shows**  $r * (k - 1) = \Lambda * (v - 1)$

$\langle \text{proof} \rangle$

**lemma** *bibd-point-occ-rep*:

**assumes**  $x \in bl$

**assumes**  $bl \in \# \mathcal{B}$

**shows**  $(\mathcal{B} - \{\# bl\}) \text{ rep } x = r - 1$

$\langle \text{proof} \rangle$

**lemma** *necess-cond-2-lhs*:  $\text{size } \{\# x \in \# (\text{mset-set } \mathcal{V} \times \# \mathcal{B}) . (\text{fst } x) \in (\text{snd } x) \#\} = v * r$

$\langle \text{proof} \rangle$

**lemma** *necess-cond-2-rhs*:  $\text{size } \{\# x \in \# (\text{mset-set } \mathcal{V} \times \# \mathcal{B}) . (\text{fst } x) \in (\text{snd } x) \#\}$



$\# \} = b * k$   
*(is size {# x ∈# (?M ×# ?B). (fst x) ∈ (snd x) #} = b\*k)*  
 ⟨proof⟩

**lemma** *necessary-condition-two*:  
 shows  $v * r = b * k$   
 ⟨proof⟩

**theorem** *admissability-conditions*:  
 $r * (k - 1) = \Lambda * (v - 1)$   
 $v * r = b * k$   
 ⟨proof⟩

### 5.2.1 BIBD Param Relationships

**lemma** *bibd-block-number*:  $b = \Lambda * v * (v - 1) \text{ div } (k * (k - 1))$   
 ⟨proof⟩

**lemma** *symmetric-condition-1*:  $\Lambda * (v - 1) = k * (k - 1) \implies b = v \wedge r = k$   
 ⟨proof⟩

**lemma** *index-lt-replication*:  $\Lambda < r$   
 ⟨proof⟩

**lemma** *index-not-zero*:  $\Lambda \geq 1$   
 ⟨proof⟩

**lemma** *r-ge-two*:  $r \geq 2$   
 ⟨proof⟩

**lemma** *block-num-gt-rep*:  $b > r$   
 ⟨proof⟩

**lemma** *bibd-subset-occ*:  
 assumes  $x \subseteq bl$  and  $bl \in \# \mathcal{B}$  and  $\text{card } x = 2$   
 shows  $\text{size } \{ \# blk \in \# (\mathcal{B} - \{ \# bl \# \}) . x \subseteq blk \# \} = \Lambda - 1$   
 ⟨proof⟩

**lemma** *necess-cond-one-param-balance*:  $b > v \implies r > k$   
 ⟨proof⟩

### 5.3 Constructing New bibd's

There are many constructions on bibd's to establish new bibds (or other types of designs). This section demonstrates this using both existing constructions, and by defining new constructions.

### 5.3.1 BIBD Complement, Multiple, Combine

**lemma** *comp-params-index-pair*:

**assumes**  $\{x, y\} \subseteq \mathcal{V}$

**assumes**  $x \neq y$

**shows**  $\mathcal{B}^C \text{ index } \{x, y\} = b + \Lambda - 2*r$

*<proof>*

**lemma** *complement-bibd-index*:

**assumes**  $ps \subseteq \mathcal{V}$

**assumes**  $\text{card } ps = 2$

**shows**  $\mathcal{B}^C \text{ index } ps = b + \Lambda - 2*r$

*<proof>*

**lemma** *complement-bibd*:

**assumes**  $k \leq v - 2$

**shows**  $\text{bibd } \mathcal{V} \mathcal{B}^C (v - k) (b + \Lambda - 2*r)$

*<proof>*

**lemma** *multiple-bibd*:  $n > 0 \implies \text{bibd } \mathcal{V} (\text{multiple-blocks } n) k (\Lambda * n)$

*<proof>*

**end**

**locale** *two-bibd-eq-points = two-t-designs-eq-points*  $\mathcal{V} \mathcal{B} k \mathcal{B}' 2 \Lambda \Lambda'$

+ *des1*:  $\text{bibd } \mathcal{V} \mathcal{B} k \Lambda$  + *des2*:  $\text{bibd } \mathcal{V} \mathcal{B}' k \Lambda'$  **for**  $\mathcal{V} \mathcal{B} k \mathcal{B}' \Lambda \Lambda'$

**begin**

**lemma** *combine-is-bibd*:  $\text{bibd } \mathcal{V}^+ \mathcal{B}^+ k (\Lambda + \Lambda')$

*<proof>*

**sublocale** *combine-bibd*:  $\text{bibd } \mathcal{V}^+ \mathcal{B}^+ k (\Lambda + \Lambda')$

*<proof>*

**end**

### 5.3.2 Derived Designs

A derived bibd takes a block from a valid bibd as the new point sets, and the intersection of that block with other blocks as it's block set

**locale** *bibd-block-transformations = bibd +*

**fixes** *block* :: 'a set (bl)

**assumes** *valid-block*:  $bl \in \# \mathcal{B}$

**begin**

**definition** *derived-blocks* :: 'a set multiset ( $(\mathcal{B}^D)$ ) **where**

$\mathcal{B}^D \equiv \{\# \text{ bl } \cap b . b \in \# (\mathcal{B} - \{\# \text{ bl} \# \}) \# \}$

**lemma** *derive-define-flip*:  $\{\# b \cap bl . b \in \# (\mathcal{B} - \{\# \text{ bl} \# \}) \# \} = \mathcal{B}^D$

*<proof>*

**lemma** *derived-points-order*:  $\text{card } bl = k$

*<proof>*

**lemma** *derived-block-num*:  $bl \in \# \mathcal{B} \implies \text{size } \mathcal{B}^D = b - 1$

*<proof>*

**lemma** *derived-is-wellformed*:  $b \in \# \mathcal{B}^D \implies b \subseteq bl$

*<proof>*

**lemma** *derived-point-subset-orig*:  $ps \subseteq bl \implies ps \subset \mathcal{V}$

*<proof>*

**lemma** *derived-obtain-orig-block*:

**assumes**  $b \in \# \mathcal{B}^D$

**obtains**  $b2$  **where**  $b = b2 \cap bl$  **and**  $b2 \in \# \text{remove1-mset } bl \mathcal{B}$

*<proof>*

**sublocale** *derived-incidence-sys*: *incidence-system*  $bl \mathcal{B}^D$

*<proof>*

**sublocale** *derived-fin-incidence-system*: *finite-incidence-system*  $bl \mathcal{B}^D$

*<proof>*

**lemma** *derived-blocks-nempty*:

**assumes**  $\bigwedge b. b \in \# \text{remove1-mset } bl \mathcal{B} \implies bl \cap b > 0$

**assumes**  $bld \in \# \mathcal{B}^D$

**shows**  $bld \neq \{\}$

*<proof>*

**lemma** *derived-is-design*:

**assumes**  $\bigwedge b. b \in \# \text{remove1-mset } bl \mathcal{B} \implies bl \cap b > 0$

**shows** *design*  $bl \mathcal{B}^D$

*<proof>*

**lemma** *derived-is-proper*:

**assumes**  $\bigwedge b. b \in \# \text{remove1-mset } bl \mathcal{B} \implies bl \cap b > 0$

**shows** *proper-design*  $bl \mathcal{B}^D$

*<proof>*

### 5.3.3 Residual Designs

Similar to derived designs, a residual design takes the complement of a block  $bl$  as it's new point set, and the complement of all other blocks with respect to  $bl$ .

**definition** *residual-blocks* :: 'a set multiset  $((\mathcal{B}^R))$  **where**  
 $\mathcal{B}^R \equiv \{\# b - bl . b \in \# (\mathcal{B} - \{\#bl\#}) \# \}$

**lemma** *residual-order*:  $\text{card } (bl^c) = v - k$   
*<proof>*

**lemma** *residual-block-num*:  $\text{size } (\mathcal{B}^R) = b - 1$   
*<proof>*

**lemma** *residual-obtain-orig-block*:  
**assumes**  $b \in \# \mathcal{B}^R$   
**obtains**  $bl2$  **where**  $b = bl2 - bl$  **and**  $bl2 \in \# \text{remove1-mset } bl \mathcal{B}$   
*<proof>*

**lemma** *residual-blocks-ss*: **assumes**  $b \in \# \mathcal{B}^R$  **shows**  $b \subseteq \mathcal{V}$   
*<proof>*

**lemma** *residual-blocks-exclude*:  $b \in \# \mathcal{B}^R \implies x \in b \implies x \notin bl$   
*<proof>*

**lemma** *residual-is-wellformed*:  $b \in \# \mathcal{B}^R \implies b \subseteq (bl^c)$   
*<proof>*

**sublocale** *residual-incidence-sys*: *incidence-system*  $bl^c \mathcal{B}^R$   
*<proof>*

**lemma** *residual-is-finite*: *finite*  $(bl^c)$   
*<proof>*

**sublocale** *residual-fin-incidence-sys*: *finite-incidence-system*  $bl^c \mathcal{B}^R$   
*<proof>*

**lemma** *residual-blocks-nempty*:  
**assumes**  $bld \in \# \mathcal{B}^R$   
**assumes** *multiplicity*  $bl = 1$   
**shows**  $bld \neq \{\}$   
*<proof>*

**lemma** *residual-is-design*: *multiplicity*  $bl = 1 \implies \text{design } (bl^c) \mathcal{B}^R$   
*<proof>*

**lemma** *residual-is-proper*:  
**assumes** *multiplicity*  $bl = 1$   
**shows** *proper-design*  $(bl^c) \mathcal{B}^R$   
*<proof>*

**end**

## 5.4 Symmetric BIBD's

Symmetric bibd's are those where the order of the design equals the number of blocks

**locale** *symmetric-bibd* = *bibd* +  
**assumes** *symmetric*:  $b = v$   
**begin**

**lemma** *rep-value-sym*:  $r = k$   
 $\langle proof \rangle$

**lemma** *symmetric-condition-2*:  $\Lambda * (v - 1) = k * (k - 1)$   
 $\langle proof \rangle$

**lemma** *sym-design-vk-gt-kl*:  
**assumes**  $k \geq \Lambda + 2$   
**shows**  $v - k > k - \Lambda$   
 $\langle proof \rangle$

**end**

**context** *bibd*  
**begin**

**lemma** *symmetric-bibdI*:  $b = v \implies \text{symmetric-bibd} \vee \mathcal{B} \ k \ \Lambda$   
 $\langle proof \rangle$

**lemma** *symmetric-bibdII*:  $\Lambda * (v - 1) = k * (k - 1) \implies \text{symmetric-bibd} \vee \mathcal{B} \ k \ \Lambda$   
 $\langle proof \rangle$

**lemma** *symmetric-not-admissible*:  $\Lambda * (v - 1) \neq k * (k - 1) \implies \neg \text{symmetric-bibd} \vee \mathcal{B} \ k \ \Lambda$   
 $\langle proof \rangle$   
**end**

**context** *symmetric-bibd*  
**begin**

### 5.4.1 Intersection Property on Symmetric BIBDs

Below is a proof of an important property on symmetric BIBD's regarding the equivalence of intersection numbers and the design index. This is an intuitive counting proof, and involved significantly more work in a formal environment. Based of Lecture Note [4]

**lemma** *intersect-mult-set-eq-block*:  
**assumes**  $blv \in \# \mathcal{B}$   
**shows**  $p \in \# \sum_{\#} \{ \# \text{ mset-set } (bl \cap blv) . bl \in \# (\mathcal{B} - \{ \# blv \# \}) \# \} \longleftrightarrow p \in blv$   
 $\langle proof \rangle$

**lemma** *intersect-mult-set-block-subset-iff*:  
**assumes**  $blv \in \# \mathcal{B}$   
**assumes**  $p \in \# \sum_{\#} \{ \# \text{ mset-set } \{ y . y \subseteq blv \cap b2 \wedge \text{card } y = 2 \} . b2 \in \# (\mathcal{B} - \{ \# blv \# \}) \# \}$

**shows**  $p \subseteq blv$   
 ⟨proof⟩

**lemma** *intersect-mult-set-block-subset-card:*

**assumes**  $blv \in \# \mathcal{B}$   
**assumes**  $p \in \# \sum_{\#} \{ \# \text{ mset-set } \{y . y \subseteq blv \cap b2 \wedge \text{card } y = 2\} . b2 \in \# (\mathcal{B} - \{\#blv\}) \# \}$   
**shows**  $\text{card } p = 2$   
 ⟨proof⟩

**lemma** *intersect-mult-set-block-with-point-exists:*

**assumes**  $blv \in \# \mathcal{B}$  **and**  $p \subseteq blv$  **and**  $\Lambda \geq 2$  **and**  $\text{card } p = 2$   
**shows**  $\exists x \in \# \text{remove1-mset } blv \mathcal{B} . p \in \# \text{ mset-set } \{y . y \subseteq blv \wedge y \subseteq x \wedge \text{card } y = 2\}$   
 ⟨proof⟩

**lemma** *intersect-mult-set-block-subset-iff-2:*

**assumes**  $blv \in \# \mathcal{B}$  **and**  $p \subseteq blv$  **and**  $\Lambda \geq 2$  **and**  $\text{card } p = 2$   
**shows**  $p \in \# \sum_{\#} \{ \# \text{ mset-set } \{y . y \subseteq blv \cap b2 \wedge \text{card } y = 2\} . b2 \in \# (\mathcal{B} - \{\#blv\}) \# \}$   
 ⟨proof⟩

**lemma** *sym-sum-mset-inter-sets-count:*

**assumes**  $blv \in \# \mathcal{B}$   
**assumes**  $p \in blv$   
**shows**  $\text{count } (\sum_{\#} \{ \# \text{ mset-set } (bl \cap blv) . bl \in \# (\mathcal{B} - \{\#blv\}) \# \}) p = r - 1$   
 (is  $\text{count } (\sum_{\#} ?M) p = r - 1$ )  
 ⟨proof⟩

**lemma** *sym-sum-mset-inter-sets-size:*

**assumes**  $blv \in \# \mathcal{B}$   
**shows**  $\text{size } (\sum_{\#} \{ \# \text{ mset-set } (bl \cap blv) . bl \in \# (\mathcal{B} - \{\#blv\}) \# \}) = k * (r - 1)$   
 (is  $\text{size } (\sum_{\#} ?M) = k * (r - 1)$ )  
 ⟨proof⟩

**lemma** *sym-sum-inter-num:*

**assumes**  $b1 \in \# \mathcal{B}$   
**shows**  $(\sum b2 \in \# (\mathcal{B} - \{\#b1\}) . b1 \mid \cap \mid b2) = k * (r - 1)$   
 ⟨proof⟩

**lemma** *sym-sum-mset-inter2-sets-count:*

**assumes**  $blv \in \# \mathcal{B}$   
**assumes**  $p \subseteq blv$   
**assumes**  $\text{card } p = 2$   
**shows**  $\text{count } (\sum_{\#} \{ \# \text{ mset-set } \{y . y \subseteq blv \cap b2 \wedge \text{card } y = 2\} . b2 \in \# (\mathcal{B} - \{\#blv\}) \# \}) p = \Lambda - 1$   
 (is  $\text{count } (\sum_{\#} ?M) p = \Lambda - 1$ )  
 ⟨proof⟩

**lemma** *sym-sum-mset-inter2-sets-size*:

**assumes**  $bl \in \# \mathcal{B}$   
**shows**  $\text{size} (\sum \# \{ \# \text{mset-set } \{y . y \subseteq bl \cap b2 \wedge \text{card } y = 2 \}. b2 \in \# (\mathcal{B} - \{ \# bl \# \}) \# \}) =$   
 $(k \text{ choose } 2) * (\Lambda - 1)$   
**is**  $\text{size} (\sum \# ?M) = (k \text{ choose } 2) * (\Lambda - 1)$   
 $\langle \text{proof} \rangle$

**lemma** *sum-choose-two-inter-num*:

**assumes**  $b1 \in \# \mathcal{B}$   
**shows**  $(\sum b2 \in \# (\mathcal{B} - \{ \# b1 \# \}). ((b1 \mid \cap \mid b2) \text{ choose } 2)) = ((\Lambda * (\Lambda - 1) \text{ div } 2)) * (v - 1)$   
 $\langle \text{proof} \rangle$

**lemma** *sym-sum-inter-num-sq*:

**assumes**  $b1 \in \# \mathcal{B}$   
**shows**  $(\sum bl \in \# (\text{remove1-mset } b1 \mathcal{B}). (b1 \mid \cap \mid bl)^2) = \Lambda^2 * (v - 1)$   
 $\langle \text{proof} \rangle$

**lemma** *sym-sum-inter-num-to-zero*:

**assumes**  $b1 \in \# \mathcal{B}$   
**shows**  $(\sum bl \in \# (\text{remove1-mset } b1 \mathcal{B}). (\text{int } (b1 \mid \cap \mid bl) - (\text{int } \Lambda))^2) = 0$   
 $\langle \text{proof} \rangle$

**theorem** *sym-block-intersections-index* [simp]:

**assumes**  $b1 \in \# \mathcal{B}$   
**assumes**  $b2 \in \# (\mathcal{B} - \{ \# b1 \# \})$   
**shows**  $b1 \mid \cap \mid b2 = \Lambda$   
 $\langle \text{proof} \rangle$

### 5.4.2 Symmetric BIBD is Simple

**lemma** *sym-block-mult-one* [simp]:

**assumes**  $bl \in \# \mathcal{B}$   
**shows** *multiplicity*  $bl = 1$   
 $\langle \text{proof} \rangle$

**end**

**sublocale** *symmetric-bibd*  $\subseteq$  *simple-design*

$\langle \text{proof} \rangle$

### 5.4.3 Residual/Derived Sym BIBD Constructions

Using the intersect result, we can reason further on residual and derived designs. Proofs based off lecture notes [4]

**locale** *symmetric-bibd-block-transformations* = *symmetric-bibd* + *bid-block-transformations*  
**begin**

**lemma** *derived-block-size* [*simp*]:  
**assumes**  $b \in \# \mathcal{B}^D$   
**shows**  $\text{card } b = \Lambda$   
 $\langle \text{proof} \rangle$

**lemma** *derived-points-index* [*simp*]:  
**assumes**  $ps \subseteq bl$   
**assumes**  $\text{card } ps = 2$   
**shows**  $\mathcal{B}^D \text{ index } ps = \Lambda - 1$   
 $\langle \text{proof} \rangle$

**lemma** *sym-derive-design-bibd*:  
**assumes**  $\Lambda > 1$   
**shows**  $\text{bibd } bl \mathcal{B}^D \Lambda (\Lambda - 1)$   
 $\langle \text{proof} \rangle$

**lemma** *residual-block-size* [*simp*]:  
**assumes**  $b \in \# \mathcal{B}^R$   
**shows**  $\text{card } b = k - \Lambda$   
 $\langle \text{proof} \rangle$

**lemma** *residual-index* [*simp*]:  
**assumes**  $ps \subseteq bl^c$   
**assumes**  $\text{card } ps = 2$   
**shows**  $(\mathcal{B}^R) \text{ index } ps = \Lambda$   
 $\langle \text{proof} \rangle$

**lemma** *sym-residual-design-bibd*:  
**assumes**  $k \geq \Lambda + 2$   
**shows**  $\text{bibd } (bl^c) \mathcal{B}^R (k - \Lambda) \Lambda$   
 $\langle \text{proof} \rangle$

**end**

## 5.5 BIBD's and Other Block Designs

BIBD's are closely related to other block designs by indirect inheritance

**sublocale**  $\text{bibd} \subseteq k\text{-}\Lambda\text{-PBD} \vee \mathcal{B} \Lambda k$   
 $\langle \text{proof} \rangle$

**lemma** *incomplete-PBD-is-bibd*:  
**assumes**  $k < \text{card } V$  **and**  $k\text{-}\Lambda\text{-PBD } V B \Lambda k$   
**shows**  $\text{bibd } V B k \Lambda$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *bibd*) *bid-to-pbdI*[*intro*]:  
**assumes**  $\Lambda = 1$   
**shows**  $k\text{-PBD} \vee \mathcal{B} k$



*<proof>*

**locale** *incomplete-PBD* = *incomplete-design* + *k- $\Lambda$ -PBD*

**sublocale** *incomplete-PBD*  $\subseteq$  *ibid*

*<proof>*

**end**

## 6 Resolvable Designs

Resolvable designs have further structure, and can be "resolved" into a set of resolution classes. A resolution class is a subset of blocks which exactly partitions the point set. Definitions based off the handbook [3] and Stinson [6]. This theory includes a proof of an alternate statement of Bose's theorem

**theory** *Resolvable-Designs* **imports** *BIBD*

**begin**

### 6.1 Resolutions and Resolution Classes

A resolution class is a partition of the point set using a set of blocks from the design. A resolution is a group of resolution classes partitioning the block collection.

**context** *incidence-system*

**begin**

**definition** *resolution-class* :: 'a set  $\Rightarrow$  bool **where**

*resolution-class*  $S \iff$  *partition-on*  $\mathcal{V} S \wedge (\forall bl \in S . bl \in\# \mathcal{B})$

**lemma** *resolution-classI* [intro]: *partition-on*  $\mathcal{V} S \implies (\bigwedge bl . bl \in S \implies bl \in\# \mathcal{B})$

$\implies$  *resolution-class*  $S$

*<proof>*

**lemma** *resolution-classD1*: *resolution-class*  $S \implies$  *partition-on*  $\mathcal{V} S$

*<proof>*

**lemma** *resolution-classD2*: *resolution-class*  $S \implies bl \in S \implies bl \in\# \mathcal{B}$

*<proof>*

**lemma** *resolution-class-empty-iff*: *resolution-class*  $\{\}$   $\iff \mathcal{V} = \{\}$

*<proof>*

**lemma** *resolution-class-complete*:  $\mathcal{V} \neq \{\} \implies \mathcal{V} \in\# \mathcal{B} \implies$  *resolution-class*  $\{\mathcal{V}\}$

*<proof>*

**lemma** *resolution-class-union*: *resolution-class*  $S \implies \bigcup S = \mathcal{V}$

*<proof>*

**lemma** (in *finite-incidence-system*) *resolution-class-finite: resolution-class S  $\implies$  finite S*  
*<proof>*

**lemma** (in *design*) *resolution-class-sum-card: resolution-class S  $\implies$  ( $\sum bl \in S . card\ bl = v$ )*  
*<proof>*

**definition** *resolution:: 'a set multiset multiset  $\implies$  bool where*  
*resolution P  $\longleftrightarrow$  partition-on-mset B P  $\wedge$  ( $\forall S \in\# P . distinct-mset S \wedge resolution-class (set-mset S)$ )*

**lemma** *resolutionI : partition-on-mset B P  $\implies$  ( $\wedge S . S \in\# P \implies distinct-mset S$ )  $\implies$*   
*( $\wedge S . S \in\# P \implies resolution-class (set-mset S)$ )  $\implies$  resolution P*  
*<proof>*

**lemma** (in *proper-design*) *resolution-blocks: distinct-mset B  $\implies$  disjoint (set-mset B)*  
 *$\implies$*   
 *$\bigcup (set-mset B) = \mathcal{V} \implies resolution \{\#B\}$*   
*<proof>*

end

## 6.2 Resolvable Design Locale

A resolvable design is one with a resolution P

**locale** *resolvable-design = design +*  
**fixes** *partition :: 'a set multiset multiset (P)*  
**assumes** *resolvable: resolution P*  
**begin**

**lemma** *resolutionD1: partition-on-mset B P*  
*<proof>*

**lemma** *resolutionD2: S  $\in\# P \implies distinct-mset S$*   
*<proof>*

**lemma** *resolutionD3: S  $\in\# P \implies resolution-class (set-mset S)$*   
*<proof>*

**lemma** *resolution-class-blocks-disjoint: S  $\in\# P \implies disjoint (set-mset S)$*   
*<proof>*

**lemma** *resolution-not-empty: B  $\neq \{\#\} \implies P \neq \{\#\}$*   
*<proof>*

**lemma** *resolution-blocks-subset*:  $S \in \# \mathcal{P} \implies S \subseteq \# \mathcal{B}$   
 ⟨*proof*⟩

**end**

**lemma** (**in** *incidence-system*) *resolvable-designI* [*intro*]: *resolution*  $\mathcal{P} \implies$  *design*  $\mathcal{V}$   
 $\mathcal{B} \implies$   
*resolvable-design*  $\mathcal{V} \mathcal{B} \mathcal{P}$   
 ⟨*proof*⟩

### 6.3 Resolvable Block Designs

An RBIBD is a resolvable BIBD - a common subclass of interest for block designs

**locale** *r-block-design* = *resolvable-design* + *block-design*

**begin**

**lemma** *resolution-class-blocks-constant-size*:  $S \in \# \mathcal{P} \implies bl \in \# S \implies \text{card } bl = k$   
 ⟨*proof*⟩

**lemma** *resolution-class-size1*:

**assumes**  $S \in \# \mathcal{P}$

**shows**  $v = k * \text{size } S$

⟨*proof*⟩

**lemma** *resolution-class-size2*:

**assumes**  $S \in \# \mathcal{P}$

**shows**  $\text{size } S = v \text{ div } k$

⟨*proof*⟩

**lemma** *resolvable-necessary-cond-v*:  $k \text{ dvd } v$

⟨*proof*⟩

**end**

**locale** *rbibd* = *r-block-design* + *bibd*

**begin**

**lemma** *resolvable-design-num-res-classes*:  $\text{size } \mathcal{P} = r$

⟨*proof*⟩

**lemma** *resolvable-necessary-cond-b*:  $r \text{ dvd } b$

⟨*proof*⟩

#### 6.3.1 Bose's Inequality

Bose's inequality is an important theorem on RBIBD's. This is a proof of an alternate statement of the thm, which does not require a linear algebraic

approach, taken directly from Stinson [6]

**theorem** *bose-inequality-alternate*:  $b \geq v + r - 1 \iff r \geq k + \Lambda$

*<proof>*

**end**

**end**

## 7 Group Divisible Designs

Definitions in this section taken from the handbook [3] and Stinson [6]

**theory** *Group-Divisible-Designs* **imports** *Resolvable-Designs*

**begin**

### 7.1 Group design

We define a group design to have an additional parameter  $\mathcal{G}$  which is a partition on the point set  $V$ . This is not defined in the handbook, but is a precursor to GDD's without index constraints

**locale** *group-design* = *proper-design* +

**fixes** *groups* :: 'a set set ( $\mathcal{G}$ )

**assumes** *group-partitions*: *partition-on*  $\mathcal{V}$   $\mathcal{G}$

**assumes** *groups-size*: *card*  $\mathcal{G} > 1$

**begin**

**lemma** *groups-not-empty*:  $\mathcal{G} \neq \{\}$

*<proof>*

**lemma** *num-groups-lt-points*: *card*  $\mathcal{G} \leq v$

*<proof>*

**lemma** *groups-disjoint*: *disjoint*  $\mathcal{G}$

*<proof>*

**lemma** *groups-disjoint-pairwise*:  $G1 \in \mathcal{G} \implies G2 \in \mathcal{G} \implies G1 \neq G2 \implies \text{disjnt } G1$   
 $G2$

*<proof>*

**lemma** *point-in-one-group*:  $x \in G1 \implies G1 \in \mathcal{G} \implies G2 \in \mathcal{G} \implies G1 \neq G2 \implies x$   
 $\notin G2$

*<proof>*

**lemma** *point-has-unique-group*:  $x \in \mathcal{V} \implies \exists! G. x \in G \wedge G \in \mathcal{G}$

*<proof>*

**lemma** *rep-number-point-group-one*:

**assumes**  $x \in \mathcal{V}$

**shows** *card*  $\{g \in \mathcal{G} . x \in g\} = 1$

*<proof>*

**lemma** *point-in-group*:  $G \in \mathcal{G} \implies x \in G \implies x \in \mathcal{V}$   
 ⟨proof⟩

**lemma** *point-subset-in-group*:  $G \in \mathcal{G} \implies ps \subseteq G \implies ps \subseteq \mathcal{V}$   
 ⟨proof⟩

**lemma** *group-subset-point-subset*:  $G \in \mathcal{G} \implies G' \subseteq G \implies ps \subseteq G' \implies ps \subseteq \mathcal{V}$   
 ⟨proof⟩

**lemma** *groups-finite*: *finite*  $\mathcal{G}$   
 ⟨proof⟩

**lemma** *group-elements-finite*:  $G \in \mathcal{G} \implies$  *finite*  $G$   
 ⟨proof⟩

**lemma** *v-equals-sum-group-sizes*:  $v = (\sum G \in \mathcal{G}. \text{card } G)$   
 ⟨proof⟩

**lemma** *gdd-min-v*:  $v \geq 2$   
 ⟨proof⟩

**lemma** *min-group-size*:  $G \in \mathcal{G} \implies \text{card } G \geq 1$   
 ⟨proof⟩

**lemma** *group-size-lt-v*:  
**assumes**  $G \in \mathcal{G}$   
**shows**  $\text{card } G < v$   
 ⟨proof⟩

### 7.1.1 Group Type

GDD's have a "type", which is defined by a sequence of group sizes  $g_i$ , and the number of groups of that size  $a_i$ :  $g_1^{a_1} g_2^{a_2} \dots g_n^{a_n}$

**definition** *group-sizes* :: *nat set* **where**  
*group-sizes*  $\equiv \{ \text{card } G \mid G . G \in \mathcal{G} \}$

**definition** *groups-of-size* :: *nat*  $\Rightarrow$  *nat* **where**  
*groups-of-size*  $g \equiv \text{card } \{ G \in \mathcal{G} . \text{card } G = g \}$

**definition** *group-type* :: (*nat*  $\times$  *nat*) *set* **where**  
*group-type*  $\equiv \{ (g, \text{groups-of-size } g) \mid g . g \in \text{group-sizes} \}$

**lemma** *group-sizes-min*:  $x \in \text{group-sizes} \implies x \geq 1$   
 ⟨proof⟩

**lemma** *group-sizes-max*:  $x \in \text{group-sizes} \implies x < v$   
 ⟨proof⟩

**lemma** *group-size-implies-group-existence*:  $x \in \text{group-sizes} \implies \exists G. G \in \mathcal{G} \wedge \text{card } G = x$   
 ⟨proof⟩

**lemma** *groups-of-size-zero*:  $\text{groups-of-size } 0 = 0$   
 ⟨proof⟩

**lemma** *groups-of-size-max*:  
**assumes**  $g \geq v$   
**shows**  $\text{groups-of-size } g = 0$   
 ⟨proof⟩

**lemma** *group-type-contained-sizes*:  $(g, a) \in \text{group-type} \implies g \in \text{group-sizes}$   
 ⟨proof⟩

**lemma** *group-type-contained-count*:  $(g, a) \in \text{group-type} \implies \text{card } \{G \in \mathcal{G} . \text{card } G = g\} = a$   
 ⟨proof⟩

**lemma** *group-card-in-sizes*:  $g \in \mathcal{G} \implies \text{card } g \in \text{group-sizes}$   
 ⟨proof⟩

**lemma** *group-card-non-zero-groups-of-size-min*:  
**assumes**  $g \in \mathcal{G}$   
**assumes**  $\text{card } g = a$   
**shows**  $\text{groups-of-size } a \geq 1$   
 ⟨proof⟩

**lemma** *elem-in-group-sizes-min-of-size*:  
**assumes**  $a \in \text{group-sizes}$   
**shows**  $\text{groups-of-size } a \geq 1$   
 ⟨proof⟩

**lemma** *group-card-non-zero-groups-of-size-max*:  
**shows**  $\text{groups-of-size } a \leq v$   
 ⟨proof⟩

**lemma** *group-card-in-type*:  $g \in \mathcal{G} \implies \exists x . (\text{card } g, x) \in \text{group-type} \wedge x \geq 1$   
 ⟨proof⟩

**lemma** *partition-groups-on-size*:  $\text{partition-on } \mathcal{G} \{\{ G \in \mathcal{G} . \text{card } G = g \} \mid g . g \in \text{group-sizes}\}$   
 ⟨proof⟩

**lemma** *group-size-partition-covers-points*:  $\bigcup (\bigcup \{\{ G \in \mathcal{G} . \text{card } G = g \} \mid g . g \in \text{group-sizes}\}) = \mathcal{V}$   
 ⟨proof⟩

**lemma** *groups-of-size-alt-def-count*:  $\text{groups-of-size } g = \text{count } \{\# \text{card } G . G \in \#$

*mset-set*  $\mathcal{G}$   $\#$  }  $g$   
 <proof>

**lemma** *v-sum-type-rep*:  $v = (\sum g \in \text{group-sizes} . g * (\text{groups-of-size } g))$   
 <proof>

**end**

### 7.1.2 Uniform Group designs

A group design requiring all groups are the same size

**locale** *uniform-group-design* = *group-design* +  
**fixes** *u-group-size* :: *nat* ( $m$ )  
**assumes** *uniform-groups*:  $G \in \mathcal{G} \implies \text{card } G = m$

**begin**

**lemma** *m-positive*:  $m \geq 1$   
 <proof>

**lemma** *uniform-groups-alt*:  $\forall G \in \mathcal{G} . \text{card } G = m$   
 <proof>

**lemma** *uniform-groups-group-sizes*: *group-sizes* =  $\{m\}$   
 <proof>

**lemma** *uniform-groups-group-size-singleton*: *is-singleton* (*group-sizes*)  
 <proof>

**lemma** *set-filter-eq-P-forall*:  $\forall x \in X . P x \implies \text{Set.filter } P X = X$   
 <proof>

**lemma** *uniform-groups-groups-of-size-m*: *groups-of-size*  $m = \text{card } \mathcal{G}$   
 <proof>

**lemma** *uniform-groups-of-size-not-m*:  $x \neq m \implies \text{groups-of-size } x = 0$   
 <proof>

**end**

## 7.2 GDD

A GDD extends a group design with an additional index parameter. Each pair of elements must occur either  $\Lambda$  times if in diff groups, or 0 times if in the same group

**locale** *GDD* = *group-design* +  
**fixes** *index* :: *int* ( $\Lambda$ )  
**assumes** *index-ge-1*:  $\Lambda \geq 1$

**assumes** *index-together*:  $G \in \mathcal{G} \implies x \in G \implies y \in G \implies x \neq y \implies \mathcal{B} \text{ index } \{x, y\} = 0$

**assumes** *index-distinct*:  $G1 \in \mathcal{G} \implies G2 \in \mathcal{G} \implies G1 \neq G2 \implies x \in G1 \implies y \in G2 \implies$

$\mathcal{B} \text{ index } \{x, y\} = \Lambda$

**begin**

**lemma** *points-sep-groups-ne*:  $G1 \in \mathcal{G} \implies G2 \in \mathcal{G} \implies G1 \neq G2 \implies x \in G1 \implies y \in G2 \implies x \neq y$

*<proof>*

**lemma** *index-together-alt-ss*:  $ps \subseteq G \implies G \in \mathcal{G} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = 0$

*<proof>*

**lemma** *index-distinct-alt-ss*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies (\bigwedge G . G \in \mathcal{G} \implies \neg ps \subseteq G) \implies$

$\mathcal{B} \text{ index } ps = \Lambda$

*<proof>*

**lemma** *gdd-index-options*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = 0 \vee \mathcal{B} \text{ index } ps = \Lambda$

*<proof>*

**lemma** *index-zero-implies-same-group*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = 0 \implies$

$\exists G \in \mathcal{G} . ps \subseteq G$  *<proof>*

**lemma** *index-zero-implies-same-group-unique*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = 0 \implies$

$\exists! G \in \mathcal{G} . ps \subseteq G$

*<proof>*

**lemma** *index-not-zero-impl-diff-group*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = \Lambda \implies$

$(\bigwedge G . G \in \mathcal{G} \implies \neg ps \subseteq G)$

*<proof>*

**lemma** *index-zero-implies-one-group*:

**assumes**  $ps \subseteq \mathcal{V}$

**and**  $\text{card } ps = 2$

**and**  $\mathcal{B} \text{ index } ps = 0$

**shows**  $\text{size } \{\#b \in \# \text{ mset-set } \mathcal{G} . ps \subseteq b\# \} = 1$

*<proof>*

**lemma** *index-distinct-group-num-alt-def*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies$

$\text{size } \{\#b \in \# \text{ mset-set } \mathcal{G} . ps \subseteq b\# \} = 0 \implies \mathcal{B} \text{ index } ps = \Lambda$

*<proof>*



**lemma** *index-non-zero-implies-no-group*:

**assumes**  $ps \subseteq \mathcal{V}$

**and**  $\text{card } ps = 2$

**and**  $\mathcal{B} \text{ index } ps = \Lambda$

**shows**  $\text{size } \{\#b \in \# \text{ mset-set } \mathcal{G} . ps \subseteq b\# \} = 0$

*<proof>*

**lemma** *gdd-index-non-zero-iff*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies$

$\mathcal{B} \text{ index } ps = \Lambda \iff \text{size } \{\#b \in \# \text{ mset-set } \mathcal{G} . ps \subseteq b\# \} = 0$

*<proof>*

**lemma** *gdd-index-zero-iff*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies$

$\mathcal{B} \text{ index } ps = 0 \iff \text{size } \{\#b \in \# \text{ mset-set } \mathcal{G} . ps \subseteq b\# \} = 1$

*<proof>*

**lemma** *points-index-upper-bound*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps \leq \Lambda$

*<proof>*

**lemma** *index-1-imp-mult-1*:

**assumes**  $\Lambda = 1$

**assumes**  $bl \in \# \mathcal{B}$

**assumes**  $\text{card } bl \geq 2$

**shows**  $\text{multiplicity } bl = 1$

*<proof>*

**lemma** *simple-if-block-size-gt-2*:

**assumes**  $\bigwedge bl . \text{card } bl \geq 2$

**assumes**  $\Lambda = 1$

**shows**  $\text{simple-design } \mathcal{V} \mathcal{B}$

*<proof>*

**end**

### 7.2.1 Sub types of GDD's

In literature, a GDD is usually defined in a number of different ways, including factors such as block size limitations

**locale**  $K\text{-}\Lambda\text{-GDD} = K\text{-block-design} + \text{GDD}$

**locale**  $k\text{-}\Lambda\text{-GDD} = \text{block-design} + \text{GDD}$

**sublocale**  $k\text{-}\Lambda\text{-GDD} \subseteq K\text{-}\Lambda\text{-GDD} \vee \mathcal{B} \{k\} \mathcal{G} \Lambda$

*<proof>*

**locale**  $K\text{-GDD} = K\text{-}\Lambda\text{-GDD} \vee \mathcal{B} \mathcal{K} \mathcal{G} 1$

**for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ ) **and** *sizes* ( $\mathcal{K}$ ) **and** *groups* ( $\mathcal{G}$ )

**locale**  $k\text{-GDD} = k\text{-}\Lambda\text{-GDD} \vee \mathcal{B} k \mathcal{G} 1$

**for** *point-set* ( $\mathcal{V}$ ) **and** *block-collection* ( $\mathcal{B}$ ) **and** *u-block-size* ( $k$ ) **and** *groups* ( $\mathcal{G}$ )

**sublocale**  $k\text{-GDD} \subseteq K\text{-GDD} \vee \mathcal{B} \{k\} \mathcal{G}$   
 ⟨proof⟩

**lemma** (in  $K\text{-GDD}$ ) *multiplicity-1*:  $bl \in \# \mathcal{B} \implies \text{card } bl \geq 2 \implies \text{multiplicity } bl = 1$   
 ⟨proof⟩

**locale**  $R\text{GDD} = \text{GDD} + \text{resolvable-design}$

### 7.3 GDD and PBD Constructions

GDD's are commonly studied alongside PBD's (pairwise balanced designs). Many constructions have been developed for designs to create a GDD from a PBD and vice versa. In particular, Wilsons Construction is a well known construction, which is formalised in this section. It should be noted that many of the more basic constructions in this section are often stated without proof/all the necessary assumptions in textbooks/course notes.

**context**  $\text{GDD}$   
**begin**

#### 7.3.1 GDD Delete Point construction

**lemma** *delete-point-index-zero*:  
**assumes**  $G \in \{g - \{x\} \mid g. g \in \mathcal{G} \wedge g \neq \{x\}\}$   
**and**  $y \in G$  **and**  $z \in G$  **and**  $z \neq y$   
**shows** (*del-point-blocks*  $x$ ) *index*  $\{y, z\} = 0$   
 ⟨proof⟩

**lemma** *delete-point-index*:  
**assumes**  $G1 \in \{g - \{x\} \mid g. g \in \mathcal{G} \wedge g \neq \{x\}\}$   
**assumes**  $G2 \in \{g - \{x\} \mid g. g \in \mathcal{G} \wedge g \neq \{x\}\}$   
**assumes**  $G1 \neq G2$  **and**  $y \in G1$  **and**  $z \in G2$   
**shows** *del-point-blocks*  $x$  *index*  $\{y, z\} = \Lambda$   
 ⟨proof⟩

**lemma** *delete-point-group-size*:  
**assumes**  $\{x\} \in \mathcal{G} \implies \text{card } \mathcal{G} > 2$   
**shows**  $1 < \text{card } \{g - \{x\} \mid g. g \in \mathcal{G} \wedge g \neq \{x\}\}$   
 ⟨proof⟩

**lemma** *GDD-by-deleting-point*:  
**assumes**  $\bigwedge bl. bl \in \# \mathcal{B} \implies x \in bl \implies 2 \leq \text{card } bl$   
**assumes**  $\{x\} \in \mathcal{G} \implies \text{card } \mathcal{G} > 2$   
**shows**  $\text{GDD} (\text{del-point } x) (\text{del-point-blocks } x) \{g - \{x\} \mid g. g \in \mathcal{G} \wedge g \neq \{x\}\} \Lambda$   
 ⟨proof⟩

**end**

**context** *K-GDD begin*

### 7.3.2 PBD construction from GDD

Two well known PBD constructions involve taking a GDD and either combining the groups and blocks to form a new block collection, or by adjoining a point

First prove that combining the groups and block set results in a constant index

**lemma** *kgdd1-points-index-group-block:*

**assumes**  $ps \subseteq \mathcal{V}$

**and**  $card\ ps = 2$

**shows**  $(\mathcal{B} + mset-set\ \mathcal{G})\ index\ ps = 1$

*<proof>*

Combining blocks and the group set forms a PBD

**lemma** *combine-block-groups-pairwise: pairwise-balance*  $\mathcal{V}\ (\mathcal{B} + mset-set\ \mathcal{G})\ 1$

*<proof>*

**lemma** *combine-block-groups-PBD:*

**assumes**  $\bigwedge G. G \in \mathcal{G} \implies card\ G \in \mathcal{K}$

**assumes**  $\bigwedge k. k \in \mathcal{K} \implies k \geq 2$

**shows** *PBD*  $\mathcal{V}\ (\mathcal{B} + mset-set\ \mathcal{G})\ \mathcal{K}$

*<proof>*

Prove adjoining a point to each group set results in a constant points index

**lemma** *kgdd1-index-adjoin-group-block:*

**assumes**  $x \notin \mathcal{V}$

**assumes**  $ps \subseteq insert\ x\ \mathcal{V}$

**assumes**  $card\ ps = 2$

**shows**  $(\mathcal{B} + mset-set\ \{insert\ x\ g\ | g. g \in \mathcal{G}\})\ index\ ps = 1$

*<proof>*

**lemma** *pairwise-by-adjoining-point:*

**assumes**  $x \notin \mathcal{V}$

**shows** *pairwise-balance*  $(add-point\ x)\ (\mathcal{B} + mset-set\ \{insert\ x\ g\ | g. g \in \mathcal{G}\})\ 1$

*<proof>*

**lemma** *PBD-by-adjoining-point:*

**assumes**  $x \notin \mathcal{V}$

**assumes**  $\bigwedge k. k \in \mathcal{K} \implies k \geq 2$

**shows** *PBD*  $(add-point\ x)\ (\mathcal{B} + mset-set\ \{insert\ x\ g\ | g. g \in \mathcal{G}\})\ (\mathcal{K} \cup \{(card\ g) + 1\ | g. g \in \mathcal{G}\})$

*<proof>*

### 7.3.3 Wilson's Construction

Wilson's construction involves the combination of multiple k-GDD's. This proof was based of Stinson [6]

**lemma** *wilsons-construction-proper:*

**assumes**  $\text{card } I = w$

**assumes**  $w > 0$

**assumes**  $\bigwedge n. n \in \mathcal{K}' \implies n \geq 2$

**assumes**  $\bigwedge B. B \in \# \mathcal{B} \implies K\text{-GDD } (B \times I) (f B) \mathcal{K}' \{ \{x\} \times I \mid x . x \in B \}$

**shows**  $\text{proper-design } (\mathcal{V} \times I) (\sum B \in \# \mathcal{B}. (f B))$  (is proper-design ?Y ?B)

*<proof>*

**lemma** *pair-construction-block-sizes:*

**assumes**  $K\text{-GDD } (B \times I) (f B) \mathcal{K}' \{ \{x\} \times I \mid x . x \in B \}$

**assumes**  $B \in \# \mathcal{B}$

**assumes**  $b \in \# (f B)$

**shows**  $\text{card } b \in \mathcal{K}'$

*<proof>*

**lemma** *wilsons-construction-index-0:*

**assumes**  $\bigwedge B. B \in \# \mathcal{B} \implies K\text{-GDD } (B \times I) (f B) \mathcal{K}' \{ \{x\} \times I \mid x . x \in B \}$

**assumes**  $G \in \{GG \times I \mid GG. GG \in \mathcal{G}\}$

**assumes**  $X \in G$

**assumes**  $Y \in G$

**assumes**  $X \neq Y$

**shows**  $(\sum \# (\text{image-mset } f \mathcal{B})) \text{ index } \{X, Y\} = 0$

*<proof>*

**lemma** *wilsons-construction-index-1:*

**assumes**  $\bigwedge B. B \in \# \mathcal{B} \implies K\text{-GDD } (B \times I) (f B) \mathcal{K}' \{ \{x\} \times I \mid x . x \in B \}$

**assumes**  $G1 \in \{G \times I \mid G. G \in \mathcal{G}\}$

**assumes**  $G2 \in \{G \times I \mid G. G \in \mathcal{G}\}$

**assumes**  $G1 \neq G2$

**and**  $(x, ix) \in G1$  **and**  $(y, iy) \in G2$

**shows**  $(\sum \# (\text{image-mset } f \mathcal{B})) \text{ index } \{(x, ix), (y, iy)\} = (1 :: \text{int})$

*<proof>*

**theorem** *Wilsons-Construction:*

**assumes**  $\text{card } I = w$

**assumes**  $w > 0$

**assumes**  $\bigwedge n. n \in \mathcal{K}' \implies n \geq 2$

**assumes**  $\bigwedge B. B \in \# \mathcal{B} \implies K\text{-GDD } (B \times I) (f B) \mathcal{K}' \{ \{x\} \times I \mid x . x \in B \}$

**shows**  $K\text{-GDD } (\mathcal{V} \times I) (\sum B \in \# \mathcal{B}. (f B)) \mathcal{K}' \{G \times I \mid G. G \in \mathcal{G}\}$

*<proof>*

**end**

**context** *pairwise-balance*

**begin**

**lemma** *PBD-by-deleting-point*:  
**assumes**  $v > 2$   
**assumes**  $\bigwedge bl . bl \in \# \mathcal{B} \implies \text{card } bl \geq 2$   
**shows** *pairwise-balance* (*del-point*  $x$ ) (*del-point-blocks*  $x$ )  $\Lambda$   
 $\langle \text{proof} \rangle$   
**end**

**context** *k-GDD*  
**begin**

**lemma** *bibd-from-kGDD*:  
**assumes**  $k > 1$   
**assumes**  $\bigwedge g . g \in \mathcal{G} \implies \text{card } g = k - 1$   
**assumes**  $x \notin \mathcal{V}$   
**shows** *bibd* (*add-point*  $x$ ) ( $\mathcal{B} + \text{mset-set } \{ \text{insert } x \ g \mid g . g \in \mathcal{G} \}$ ) ( $k$ )  $1$   
 $\langle \text{proof} \rangle$

**end**

**context** *PBD*  
**begin**

**lemma** *pbd-points-index1*:  $ps \subseteq \mathcal{V} \implies \text{card } ps = 2 \implies \mathcal{B} \text{ index } ps = 1$   
 $\langle \text{proof} \rangle$

**lemma** *pbd-index1-points-imply-unique-block*:  
**assumes**  $b1 \in \# \mathcal{B}$  **and**  $b2 \in \# \mathcal{B}$  **and**  $b1 \neq b2$   
**assumes**  $x \neq y$  **and**  $\{x, y\} \subseteq b1$  **and**  $x \in b2$   
**shows**  $y \notin b2$   
 $\langle \text{proof} \rangle$

**lemma** *strong-delete-point-groups-index-zero*:  
**assumes**  $G \in \{b - \{x\} \mid b . b \in \# \mathcal{B} \wedge x \in b\}$   
**assumes**  $xa \in G$  **and**  $y \in G$  **and**  $xa \neq y$   
**shows** (*str-del-point-blocks*  $x$ ) *index*  $\{xa, y\} = 0$   
 $\langle \text{proof} \rangle$

**lemma** *strong-delete-point-groups-index-one*:  
**assumes**  $G1 \in \{b - \{x\} \mid b . b \in \# \mathcal{B} \wedge x \in b\}$   
**assumes**  $G2 \in \{b - \{x\} \mid b . b \in \# \mathcal{B} \wedge x \in b\}$   
**assumes**  $G1 \neq G2$  **and**  $xa \in G1$  **and**  $y \in G2$   
**shows** (*str-del-point-blocks*  $x$ ) *index*  $\{xa, y\} = 1$   
 $\langle \text{proof} \rangle$

**lemma** *blocks-with-x-partition*:  
**assumes**  $x \in \mathcal{V}$   
**shows** *partition-on* ( $\mathcal{V} - \{x\}$ )  $\{b - \{x\} \mid b . b \in \# \mathcal{B} \wedge x \in b\}$   
 $\langle \text{proof} \rangle$

```

lemma KGDD-by-deleting-point:
  assumes  $x \in \mathcal{V}$ 
  assumes  $\mathcal{B}$  rep  $x < b$ 
  assumes  $\mathcal{B}$  rep  $x > 1$ 
  shows  $K$ -GDD (del-point  $x$ ) (str-del-point-blocks  $x$ )  $\mathcal{K} \{ b - \{x\} \mid b . b \in \# \mathcal{B} \wedge$ 
 $x \in b \}$ 
  <proof>

```

```

lemma card-singletons-eq:  $\text{card} \{ \{a\} \mid a . a \in A \} = \text{card } A$ 
  <proof>

```

```

lemma KGDD-from-PBD:  $K$ -GDD  $\vee \mathcal{B} \mathcal{K} \{ \{x\} \mid x . x \in \mathcal{V} \}$ 
  <proof>

```

**end**

**context** *bibd*

**begin**

**lemma** *kGDD-from-bibd*:

**assumes**  $\Lambda = 1$

**assumes**  $x \in \mathcal{V}$

**shows**  $k$ -GDD (*del-point*  $x$ ) (*str-del-point-blocks*  $x$ )  $\mathcal{K} \{ b - \{x\} \mid b . b \in \# \mathcal{B} \wedge x$   
 $\in b \}$   
*<proof>*

**end**

**end**

## 8 Graphs and Designs

There are many links between graphs and design - most fundamentally that graphs are designs

```

theory Designs-And-Graphs imports Block-Designs Graph-Theory.Digraph Graph-Theory.Digraph-Component
begin

```

### 8.1 Non-empty digraphs

First, we define the concept of a non-empty digraph. This mirrors the idea of a "proper design" in the design theory library

```

locale non-empty-digraph = wf-digraph +
  assumes arcs-not-empty:  $\text{arcs } G \neq \{ \}$ 

```

**begin**

```

lemma verts-not-empty:  $\text{verts } G \neq \{ \}$ 
  <proof>

```

**end**

## 8.2 Arcs to Blocks

A digraph uses a pair of points to define an ordered edge. In the case of simple graphs, both possible orderings will be in the arcs set. Blocks are inherently unordered, and as such a method is required to convert between the two representations

**context** *graph*  
**begin**

**definition** *arc-to-block* :: 'b  $\Rightarrow$  'a set **where**  
*arc-to-block* e  $\equiv$  {tail G e, head G e}

**lemma** *arc-to-block-to-ends*: {fst (arc-to-ends G e), snd (arc-to-ends G e)} =  
*arc-to-block* e  
*<proof>*

**lemma** *arc-to-block-to-ends-swap*: {snd (arc-to-ends G e), fst (arc-to-ends G e)}  
= *arc-to-block* e  
*<proof>*

**lemma** *arc-to-ends-to-block*: *arc-to-block* e = {x, y}  $\implies$   
arc-to-ends G e = (x, y)  $\vee$  arc-to-ends G e = (y, x)  
*<proof>*

**lemma** *arc-to-block-sym*: arc-to-ends G e1 = (u, v)  $\implies$  arc-to-ends G e2 = (v, u)  
 $\implies$   
arc-to-block e1 = arc-to-block e2  
*<proof>*

**definition** *arcs-blocks* :: 'a set multiset **where**  
*arcs-blocks*  $\equiv$  mset-set (arc-to-block ' (arcs G))

**lemma** *arcs-blocks-ends*: (x, y)  $\in$  arcs-ends G  $\implies$  {x, y}  $\in$  # arcs-blocks  
*<proof>*

**lemma** *arc-ends-blocks-subset*: E  $\subseteq$  arcs G  $\implies$  (x, y)  $\in$  ((arc-to-ends G) ' E)  $\implies$   
{x, y}  $\in$  (arc-to-block ' E)  
*<proof>*

**lemma** *arc-blocks-end-subset*: **assumes** E  $\subseteq$  arcs G **and** {x, y}  $\in$  (arc-to-block ' E)  
**shows** (x, y)  $\in$  ((arc-to-ends G) ' E)  $\vee$  (y, x)  $\in$  ((arc-to-ends G) ' E)  
*<proof>*

**lemma** *arcs-ends-blocks*: {x, y}  $\in$  # arcs-blocks  $\implies$  (x, y)  $\in$  arcs-ends G  $\wedge$  (y, x)  $\in$  arcs-ends G

*<proof>*

**lemma** *arcs-blocks-iff*:  $\{x, y\} \in \# \text{ arcs-blocks} \longleftrightarrow (x, y) \in \text{arcs-ends } G \wedge (y, x) \in \text{arcs-ends } G$   
*<proof>*

**lemma** *arcs-ends-wf*:  $(x, y) \in \text{arcs-ends } G \implies x \in \text{verts } G \wedge y \in \text{verts } G$   
*<proof>*

**lemma** *arcs-blocks-elem*:  $bl \in \# \text{ arcs-blocks} \implies \exists x y . bl = \{x, y\}$   
*<proof>*

**lemma** *arcs-ends-blocks-wf*:  
**assumes**  $bl \in \# \text{ arcs-blocks}$   
**shows**  $bl \subseteq \text{verts } G$   
*<proof>*

**lemma** *arcs-blocks-simple*:  $bl \in \# \text{ arcs-blocks} \implies \text{count } (\text{arcs-blocks}) \text{ } bl = 1$   
*<proof>*

**lemma** *arcs-blocks-ne*:  $\text{arcs } G \neq \{\} \implies \text{arcs-blocks} \neq \{\#\}$   
*<proof>*

**end**

### 8.3 Graphs are designs

Prove that a graph is a number of different types of designs

**sublocale** *graph*  $\subseteq$  *design* *verts* *G* *arcs-blocks*  
*<proof>*

**sublocale** *graph*  $\subseteq$  *simple-design* *verts* *G* *arcs-blocks*  
*<proof>*

**locale** *non-empty-graph* = *graph* + *non-empty-digraph*

**sublocale** *non-empty-graph*  $\subseteq$  *proper-design* *verts* *G* *arcs-blocks*  
*<proof>*

**lemma** (*in graph*) *graph-block-size*: **assumes**  $bl \in \# \text{ arcs-blocks}$  **shows**  $\text{card } bl = 2$   
*<proof>*

**sublocale** *non-empty-graph*  $\subseteq$  *block-design* *verts* *G* *arcs-blocks* 2  
*<proof>*



## 8.4 R-regular graphs

To reason on r-regular graphs and their link to designs, we require a number of extensions to lemmas reasoning around the degrees of vertices

**context** *sym-digraph*  
**begin**

**lemma** *in-out-arcs-reflexive*:  $v \in \text{verts } G \implies (e \in (\text{in-arcs } G v) \implies \exists e' . (e' \in (\text{out-arcs } G v) \wedge \text{head } G e' = \text{tail } G e))$   
*<proof>*

**lemma** *out-in-arcs-reflexive*:  $v \in \text{verts } G \implies (e \in (\text{out-arcs } G v) \implies \exists e' . (e' \in (\text{in-arcs } G v) \wedge \text{tail } G e' = \text{head } G e))$   
*<proof>*

**end**

**context** *nomulti-digraph*  
**begin**

**lemma** *in-arcs-single-per-vert*:  
**assumes**  $v \in \text{verts } G$  **and**  $u \in \text{verts } G$   
**assumes**  $e1 \in \text{in-arcs } G v$  **and**  $e2 \in \text{in-arcs } G v$   
**assumes**  $\text{tail } G e1 = u$  **and**  $\text{tail } G e2 = u$   
**shows**  $e1 = e2$   
*<proof>*

**lemma** *out-arcs-single-per-vert*:  
**assumes**  $v \in \text{verts } G$  **and**  $u \in \text{verts } G$   
**assumes**  $e1 \in \text{out-arcs } G v$  **and**  $e2 \in \text{out-arcs } G v$   
**assumes**  $\text{head } G e1 = u$  **and**  $\text{head } G e2 = u$   
**shows**  $e1 = e2$   
*<proof>*

**end**

Some helpers on the transformation arc definition

**context** *graph*  
**begin**

**lemma** *arc-to-block-is-inj-in-arcs*: *inj-on arc-to-block (in-arcs G v)*  
*<proof>*

**lemma** *arc-to-block-is-inj-out-arcs*: *inj-on arc-to-block (out-arcs G v)*  
*<proof>*

**lemma** *in-out-arcs-reflexive-uniq*:  $v \in \text{verts } G \implies (e \in (\text{in-arcs } G v) \implies \exists! e' . (e' \in (\text{out-arcs } G v) \wedge \text{head } G e' = \text{tail } G e))$   
*<proof>*

**lemma** *out-in-arcs-reflexive-uniq*:  $v \in \text{verts } G \implies e \in (\text{out-arcs } G \ v) \implies \exists! e' . (e' \in (\text{in-arcs } G \ v) \wedge \text{tail } G \ e' = \text{head } G \ e)$   
 ⟨proof⟩

**lemma** *in-eq-out-arc-ends*:  $(u, v) \in ((\text{arc-to-ends } G) \text{ ' } (\text{in-arcs } G \ v)) \iff (v, u) \in ((\text{arc-to-ends } G) \text{ ' } (\text{out-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *in-degree-eq-card-arc-ends*:  $\text{in-degree } G \ v = \text{card } ((\text{arc-to-ends } G) \text{ ' } (\text{in-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *in-degree-eq-card-arc-blocks*:  $\text{in-degree } G \ v = \text{card } (\text{arc-to-block} \text{ ' } (\text{in-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *out-degree-eq-card-arc-blocks*:  $\text{out-degree } G \ v = \text{card } (\text{arc-to-block} \text{ ' } (\text{out-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *out-degree-eq-card-arc-ends*:  $\text{out-degree } G \ v = \text{card } ((\text{arc-to-ends } G) \text{ ' } (\text{out-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *bij-betw-in-out-arcs*:  $\text{bij-betw } (\lambda (u, v) . (v, u)) ((\text{arc-to-ends } G) \text{ ' } (\text{in-arcs } G \ v)) ((\text{arc-to-ends } G) \text{ ' } (\text{out-arcs } G \ v))$   
 ⟨proof⟩

**lemma** *in-eq-out-degree*:  $\text{in-degree } G \ v = \text{out-degree } G \ v$   
 ⟨proof⟩

**lemma** *in-out-arcs-blocks*:  $\text{arc-to-block} \text{ ' } (\text{in-arcs } G \ v) = \text{arc-to-block} \text{ ' } (\text{out-arcs } G \ v)$   
 ⟨proof⟩

**end**

A regular digraph is defined as one where the in degree equals the out degree which in turn equals some fixed integer r

**locale** *regular-digraph* = *wf-digraph* +  
**fixes**  $r :: \text{nat}$   
**assumes** *in-deg-r*:  $v \in \text{verts } G \implies \text{in-degree } G \ v = r$   
**assumes** *out-deg-r*:  $v \in \text{verts } G \implies \text{out-degree } G \ v = r$

**locale** *regular-graph* = *graph* + *regular-digraph*  
**begin**

**lemma** *rep-vertices-in-blocks* [*simp*]:  
**assumes**  $x \in \text{verts } G$   
**shows**  $\text{size } \{\# e \in \# \text{ arcs-blocks } . x \in e \# \} = r$   
*<proof>*

**end**

Intro rules for regular graphs

**lemma** *graph-in-degree-r-imp-reg*[*intro*]: **assumes** *graph*  $G$   
**assumes**  $(\bigwedge v . v \in (\text{verts } G) \implies \text{in-degree } G v = r)$   
**shows** *regular-graph*  $G r$   
*<proof>*

**lemma** *graph-out-degree-r-imp-reg*[*intro*]: **assumes** *graph*  $G$   
**assumes**  $(\bigwedge v . v \in (\text{verts } G) \implies \text{out-degree } G v = r)$   
**shows** *regular-graph*  $G r$   
*<proof>*

Regular graphs (non-empty) can be shown to be a constant rep design

**locale** *non-empty-regular-graph* = *regular-graph* + *non-empty-digraph*

**sublocale** *non-empty-regular-graph*  $\subseteq$  *non-empty-graph*  
*<proof>*

**sublocale** *non-empty-regular-graph*  $\subseteq$  *constant-rep-design* *verts*  $G$  *arcs-blocks*  $r$   
*<proof>*

**end**

## 9 Sub-designs

Sub designs are a relationship between two designs using the subset and submultiset relations This theory defines the concept at the incidence system level, before extending to defining on well defined designs

**theory** *Sub-Designs* **imports** *Design-Operations*  
**begin**

### 9.1 Sub-system and Sub-design Locales

**locale** *sub-set-system* = *incidence-system*  $\mathcal{V}$   $\mathcal{B}$   
**for**  $\mathcal{U}$  **and**  $\mathcal{A}$  **and**  $\mathcal{V}$  **and**  $\mathcal{B}$  +  
**assumes** *points-subset*:  $\mathcal{U} \subseteq \mathcal{V}$   
**assumes** *blocks-subset*:  $\mathcal{A} \subseteq \# \mathcal{B}$   
**begin**

**lemma** *sub-points*:  $x \in \mathcal{U} \implies x \in \mathcal{V}$   
*<proof>*

**lemma** *sub-blocks*:  $bl \in\# \mathcal{A} \implies bl \in\# \mathcal{B}$   
*<proof>*

**lemma** *sub-blocks-count*:  $count \mathcal{A} b \leq count \mathcal{B} b$   
*<proof>*

**end**

**locale** *sub-incidence-system* = *sub-set-system* + *ins: incidence-system*  $\mathcal{U} \mathcal{A}$

**locale** *sub-design* = *sub-incidence-system* + *des: design*  $\mathcal{V} \mathcal{B}$   
**begin**

**lemma** *sub-non-empty-blocks*:  $A \in\# \mathcal{A} \implies A \neq \{\}$   
*<proof>*

**sublocale** *sub-des: design*  $\mathcal{U} \mathcal{A}$   
*<proof>*

**end**

**locale** *proper-sub-set-system* = *incidence-system*  $\mathcal{V} \mathcal{B}$   
**for**  $\mathcal{U}$  **and**  $\mathcal{A}$  **and**  $\mathcal{V}$  **and**  $\mathcal{B}$  +  
**assumes** *points-psubset*:  $\mathcal{U} \subset \mathcal{V}$   
**assumes** *blocks-subset*:  $\mathcal{A} \subseteq\# \mathcal{B}$   
**begin**

**lemma** *point-sets-ne*:  $\mathcal{U} \neq \mathcal{V}$   
*<proof>*

**end**

**sublocale** *proper-sub-set-system*  $\subseteq$  *sub-set-system*  
*<proof>*

**context** *sub-set-system*  
**begin**

**lemma** *sub-is-proper*:  $\mathcal{U} \neq \mathcal{V} \implies$  *proper-sub-set-system*  $\mathcal{U} \mathcal{A} \mathcal{V} \mathcal{B}$   
*<proof>*

**end**

**locale** *proper-sub-incidence-system* = *proper-sub-set-system* + *ins: incidence-system*  
 $\mathcal{U} \mathcal{A}$

**sublocale** *proper-sub-incidence-system*  $\subseteq$  *sub-incidence-system*  
*<proof>*

```

context sub-incidence-system
begin
lemma sub-is-proper:  $\mathcal{U} \neq \mathcal{V} \implies \text{proper-sub-incidence-system } \mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$ 
  <proof>

end

locale proper-sub-design = proper-sub-incidence-system + des: design  $\mathcal{V} \ \mathcal{B}$ 

sublocale proper-sub-design  $\subseteq$  sub-design
  <proof>

context sub-design
begin
lemma sub-is-proper:  $\mathcal{U} \neq \mathcal{V} \implies \text{proper-sub-design } \mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$ 
  <proof>

end

lemma ss-proper-implies-sub [intro]: proper-sub-set-system  $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B} \implies \text{sub-set-system}$ 
 $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$ 
  <proof>

lemma sub-ssI [intro!]: incidence-system  $\mathcal{V} \ \mathcal{B} \implies \mathcal{U} \subseteq \mathcal{V} \implies \mathcal{A} \subseteq\# \mathcal{B} \implies$ 
sub-set-system  $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$ 
  <proof>

lemma sub-ss-equality:
  assumes sub-set-system  $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$ 
  and sub-set-system  $\mathcal{V} \ \mathcal{B} \ \mathcal{U} \ \mathcal{A}$ 
  shows  $\mathcal{U} = \mathcal{V}$  and  $\mathcal{A} = \mathcal{B}$ 
  <proof>

```

## 9.2 Reasoning on Sub-designs

### 9.2.1 Reasoning on Incidence Sys property relationships

```

context sub-incidence-system
begin

lemma sub-sys-block-sizes: ins.sys-block-sizes  $\subseteq$  sys-block-sizes
  <proof>

lemma sub-point-rep-number-le:  $x \in \mathcal{U} \implies \mathcal{A} \text{ rep } x \leq \mathcal{B} \text{ rep } x$ 
  <proof>

lemma sub-point-index-le:  $ps \subseteq \mathcal{U} \implies \mathcal{A} \text{ index } ps \leq \mathcal{B} \text{ index } ps$ 
  <proof>

lemma sub-sys-intersection-numbers: ins.intersection-numbers  $\subseteq$  intersection-numbers

```

*<proof>*

**end**

## 9.2.2 Reasoning on Incidence Sys/Design operations

**context** *incidence-system*

**begin**

**lemma** *sub-set-sysI*[*intro*]:  $\mathcal{U} \subseteq \mathcal{V} \implies \mathcal{A} \subseteq \# \mathcal{B} \implies \text{sub-set-system } \mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$

*<proof>*

**lemma** *sub-inc-sysI*[*intro*]: *incidence-system*  $\mathcal{U} \ \mathcal{A} \implies \mathcal{U} \subseteq \mathcal{V} \implies \mathcal{A} \subseteq \# \mathcal{B} \implies$   
*sub-incidence-system*  $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$

*<proof>*

**lemma** *multiple-orig-sub-system*:

**assumes**  $n > 0$

**shows** *sub-incidence-system*  $\mathcal{V} \ \mathcal{B} \ \mathcal{V} \ (\text{multiple-blocks } n)$

*<proof>*

**lemma** *add-point-sub-sys*: *sub-incidence-system*  $\mathcal{V} \ \mathcal{B} \ (\text{add-point } p) \ \mathcal{B}$

*<proof>*

**lemma** *strong-del-point-sub-sys*: *sub-incidence-system*  $(\text{del-point } p) \ (\text{str-del-point-blocks } p) \ \mathcal{V} \ \mathcal{B}$

*<proof>*

**lemma** *add-block-sub-sys*: *sub-incidence-system*  $\mathcal{V} \ \mathcal{B} \ (\mathcal{V} \cup b) \ (\text{add-block } b)$

*<proof>*

**lemma** *delete-block-sub-sys*: *sub-incidence-system*  $\mathcal{V} \ (\text{del-block } b) \ \mathcal{V} \ \mathcal{B}$

*<proof>*

**end**

**lemma** (*in two-set-systems*) *combine-sub-sys*: *sub-incidence-system*  $\mathcal{V} \ \mathcal{B} \ \mathcal{V}^+ \ \mathcal{B}^+$

*<proof>*

**lemma** (*in two-set-systems*) *combine-sub-sys-alt*: *sub-incidence-system*  $\mathcal{V}' \ \mathcal{B}' \ \mathcal{V}^+ \ \mathcal{B}^+$

*<proof>*

**context** *design*

**begin**

**lemma** *sub-designI* [*intro*]: *design*  $\mathcal{U} \ \mathcal{A} \implies \text{sub-incidence-system } \mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B} \implies$   
*sub-design*  $\mathcal{U} \ \mathcal{A} \ \mathcal{V} \ \mathcal{B}$

*<proof>*

**lemma** *sub-designII* [intro]: *design*  $\mathcal{U}$   $\mathcal{A} \implies$  *sub-incidence-system*  $\mathcal{V}$   $\mathcal{B}$   $\mathcal{U}$   $\mathcal{A} \implies$   
*sub-design*  $\mathcal{V}$   $\mathcal{B}$   $\mathcal{U}$   $\mathcal{A}$   
 ⟨proof⟩

**lemma** *multiple-orig-sub-des*:  
**assumes**  $n > 0$   
**shows** *sub-design*  $\mathcal{V}$   $\mathcal{B}$   $\mathcal{V}$  (*multiple-blocks*  $n$ )  
 ⟨proof⟩

**lemma** *add-point-sub-des*: *sub-design*  $\mathcal{V}$   $\mathcal{B}$  (*add-point*  $p$ )  $\mathcal{B}$   
 ⟨proof⟩

**lemma** *strong-del-point-sub-des*: *sub-design* (*del-point*  $p$ ) (*str-del-point-blocks*  $p$ )  $\mathcal{V}$   
 $\mathcal{B}$   
 ⟨proof⟩

**lemma** *add-block-sub-des*: *finite*  $b \implies b \neq \{\}$   $\implies$  *sub-design*  $\mathcal{V}$   $\mathcal{B}$  ( $\mathcal{V} \cup b$ ) (*add-block*  
 $b$ )  
 ⟨proof⟩

**lemma** *delete-block-sub-des*: *sub-design*  $\mathcal{V}$  (*del-block*  $b$ )  $\mathcal{V}$   $\mathcal{B}$   
 ⟨proof⟩

end

**lemma** (in *two-designs*) *combine-sub-des*: *sub-design*  $\mathcal{V}$   $\mathcal{B}$   $\mathcal{V}^+$   $\mathcal{B}^+$   
 ⟨proof⟩

**lemma** (in *two-designs*) *combine-sub-des-alt*: *sub-design*  $\mathcal{V}'$   $\mathcal{B}'$   $\mathcal{V}^+$   $\mathcal{B}^+$   
 ⟨proof⟩

end

## 10 Design Isomorphisms

**theory** *Design-Isomorphisms* **imports** *Design-Basics* *Sub-Designs*  
**begin**

### 10.1 Images of Set Systems

We loosely define the concept of taking the "image" of a set system, as done in isomorphisms. Note that this is not based off mathematical theory, but is for ease of notation

**definition** *blocks-image* :: ' $a$  set multiset  $\Rightarrow$  ( $'a \Rightarrow 'b$ )  $\Rightarrow$  ' $b$  set multiset **where**  
*blocks-image*  $B$   $f \equiv$  *image-mset* ( $(\cdot) f$ )  $B$

**lemma** *image-block-set-constant-size*: *size* ( $B$ ) = *size* (*blocks-image*  $B$   $f$ )

*<proof>*

**lemma** (in *incidence-system*) *image-set-system-wellformed*:  
*incidence-system* ( $f \text{ ' } \mathcal{V}$ ) (*blocks-image*  $\mathcal{B}$   $f$ )  
*<proof>*

**lemma** (in *finite-incidence-system*) *image-set-system-finite*:  
*finite-incidence-system* ( $f \text{ ' } \mathcal{V}$ ) (*blocks-image*  $\mathcal{B}$   $f$ )  
*<proof>*

## 10.2 Incidence System Isomorphisms

Isomorphism's are defined by the Handbook of Combinatorial Designs [3]

**locale** *incidence-system-isomorphism* = *source*: *incidence-system*  $\mathcal{V}$   $\mathcal{B}$  + *target*:  
*incidence-system*  $\mathcal{V}'$   $\mathcal{B}'$   
**for**  $\mathcal{V}$  **and**  $\mathcal{B}$  **and**  $\mathcal{V}'$  **and**  $\mathcal{B}'$  + **fixes** *bij-map* ( $\pi$ )  
**assumes** *bij*: *bij-betw*  $\pi$   $\mathcal{V}$   $\mathcal{V}'$   
**assumes** *block-img*: *image-mset* ( $(\cdot)$   $\pi$ )  $\mathcal{B} = \mathcal{B}'$   
**begin**

**lemma** *iso-eq-order*: *card*  $\mathcal{V} = \text{card } \mathcal{V}'$   
*<proof>*

**lemma** *iso-eq-block-num*: *size*  $\mathcal{B} = \text{size } \mathcal{B}'$   
*<proof>*

**lemma** *iso-block-img-alt-rep*:  $\{\# \pi \text{ ' } bl . bl \in \# \mathcal{B} \#\} = \mathcal{B}'$   
*<proof>*

**lemma** *inv-iso-block-img*: *image-mset* ( $(\cdot)$  (*inv-into*  $\mathcal{V}$   $\pi$ ))  $\mathcal{B}' = \mathcal{B}$   
*<proof>*

**lemma** *inverse-incidence-sys-iso*: *incidence-system-isomorphism*  $\mathcal{V}'$   $\mathcal{B}'$   $\mathcal{V}$   $\mathcal{B}$  (*inv-into*  
 $\mathcal{V}$   $\pi$ )  
*<proof>*

**lemma** *iso-points-map*:  $\pi \text{ ' } \mathcal{V} = \mathcal{V}'$   
*<proof>*

**lemma** *iso-points-inv-map*: (*inv-into*  $\mathcal{V}$   $\pi$ ) \text{ ' }  $\mathcal{V}' = \mathcal{V}$   
*<proof>*

**lemma** *iso-points-ss-card*:  
**assumes**  $ps \subseteq \mathcal{V}$   
**shows** *card*  $ps = \text{card } (\pi \text{ ' } ps)$   
*<proof>*

**lemma** *iso-block-in*:  $bl \in \# \mathcal{B} \implies (\pi \text{ ' } bl) \in \# \mathcal{B}'$   
*<proof>*



**lemma** *iso-inv-block-in*:  $x \in \# \mathcal{B}' \implies x \in (\cdot) \pi \text{ ' set-mset } \mathcal{B}$   
 ⟨proof⟩

**lemma** *iso-img-block-orig-exists*:  $x \in \# \mathcal{B}' \implies \exists bl . bl \in \# \mathcal{B} \wedge x = \pi \text{ ' } bl$   
 ⟨proof⟩

**lemma** *iso-blocks-map-inj*:  $x \in \# \mathcal{B} \implies y \in \# \mathcal{B} \implies \pi \text{ ' } x = \pi \text{ ' } y \implies x = y$   
 ⟨proof⟩

**lemma** *iso-bij-betwn-block-sets*:  $\text{bij-betw } ((\cdot) \pi) (\text{set-mset } \mathcal{B}) (\text{set-mset } \mathcal{B}')$   
 ⟨proof⟩

**lemma** *iso-bij-betwn-block-sets-inv*:  $\text{bij-betw } ((\cdot) (\text{inv-into } \mathcal{V} \pi)) (\text{set-mset } \mathcal{B}') (\text{set-mset } \mathcal{B})$   
 ⟨proof⟩

**lemma** *iso-bij-betw-individual-blocks*:  $bl \in \# \mathcal{B} \implies \text{bij-betw } \pi \text{ } bl (\pi \text{ ' } bl)$   
 ⟨proof⟩

**lemma** *iso-bij-betw-individual-blocks-inv*:  $bl \in \# \mathcal{B} \implies \text{bij-betw } (\text{inv-into } \mathcal{V} \pi) (\pi \text{ ' } bl) bl$   
 ⟨proof⟩

**lemma** *iso-bij-betw-individual-blocks-inv-alt*:  
 $bl \in \# \mathcal{B}' \implies \text{bij-betw } (\text{inv-into } \mathcal{V} \pi) bl ((\text{inv-into } \mathcal{V} \pi) \text{ ' } bl)$   
 ⟨proof⟩

**lemma** *iso-inv-block-in-alt*:  $(\pi \text{ ' } bl) \in \# \mathcal{B}' \implies bl \subseteq \mathcal{V} \implies bl \in \# \mathcal{B}$   
 ⟨proof⟩

**lemma** *iso-img-block-not-in*:  
 assumes  $x \notin \# \mathcal{B}$   
 assumes  $x \subseteq \mathcal{V}$   
 shows  $(\pi \text{ ' } x) \notin \# \mathcal{B}'$   
 ⟨proof⟩

**lemma** *iso-block-multiplicity*:  
 assumes  $bl \subseteq \mathcal{V}$   
 shows  $\text{source.multiplicity } bl = \text{target.multiplicity } (\pi \text{ ' } bl)$   
 ⟨proof⟩

**lemma** *iso-point-in-block-img-iff*:  $p \in \mathcal{V} \implies bl \in \# \mathcal{B} \implies p \in bl \iff (\pi \text{ } p) \in (\pi \text{ ' } bl)$   
 ⟨proof⟩

**lemma** *iso-point-subset-block-iff*:  $p \subseteq \mathcal{V} \implies bl \in \# \mathcal{B} \implies p \subseteq bl \iff (\pi \text{ ' } p) \subseteq (\pi \text{ ' } bl)$   
 ⟨proof⟩

**lemma** *iso-is-image-block*:  $\mathcal{B}' = \text{blocks-image } \mathcal{B} \pi$   
 ⟨proof⟩

**end**

### 10.3 Design Isomorphisms

Apply the concept of isomorphisms to designs only

**locale** *design-isomorphism* = *incidence-system-isomorphism*  $\mathcal{V} \mathcal{B} \mathcal{V}' \mathcal{B}' \pi$  + *source*:  
*design*  $\mathcal{V} \mathcal{B}$  +  
*target*: *design*  $\mathcal{V}' \mathcal{B}'$  for  $\mathcal{V}$  and  $\mathcal{B}$  and  $\mathcal{V}'$  and  $\mathcal{B}'$  and *bij-map* ( $\pi$ )

**context** *design-isomorphism*  
**begin**

**lemma** *inverse-design-isomorphism*: *design-isomorphism*  $\mathcal{V}' \mathcal{B}' \mathcal{V} \mathcal{B}$  (*inv-into*  $\mathcal{V} \pi$ )  
 ⟨proof⟩

**end**

#### 10.3.1 Isomorphism Operation

Define the concept of isomorphic designs outside the scope of locale

**definition** *isomorphic-designs* (*infixl*  $\cong_D$  50) **where**  
 $\mathcal{D} \cong_D \mathcal{D}' \longleftrightarrow (\exists \pi . \text{design-isomorphism } (\text{fst } \mathcal{D}) (\text{snd } \mathcal{D}) (\text{fst } \mathcal{D}') (\text{snd } \mathcal{D}') \pi)$

**lemma** *isomorphic-designs-symmetric*:  $(\mathcal{V}, \mathcal{B}) \cong_D (\mathcal{V}', \mathcal{B}') \implies (\mathcal{V}', \mathcal{B}') \cong_D (\mathcal{V}, \mathcal{B})$   
 ⟨proof⟩

**lemma** *isomorphic-designs-implies-bij*:  $(\mathcal{V}, \mathcal{B}) \cong_D (\mathcal{V}', \mathcal{B}') \implies \exists \pi . \text{bij-betw } \pi \mathcal{V} \mathcal{V}'$   
 ⟨proof⟩

**lemma** *isomorphic-designs-implies-block-map*:  $(\mathcal{V}, \mathcal{B}) \cong_D (\mathcal{V}', \mathcal{B}') \implies \exists \pi . \text{image-mset } ((\cdot) \pi) \mathcal{B} = \mathcal{B}'$   
 ⟨proof⟩

**context** *design*  
**begin**

**lemma** *isomorphic-designsI* [*intro*]: *design*  $\mathcal{V}' \mathcal{B}' \implies \text{bij-betw } \pi \mathcal{V} \mathcal{V}' \implies \text{image-mset } ((\cdot) \pi) \mathcal{B} = \mathcal{B}'$   
 $\implies (\mathcal{V}, \mathcal{B}) \cong_D (\mathcal{V}', \mathcal{B}')$   
 ⟨proof⟩

**lemma** *eq-designs-isomorphic*:  
**assumes**  $\mathcal{V} = \mathcal{V}'$   
**assumes**  $\mathcal{B} = \mathcal{B}'$

**shows**  $(\mathcal{V}, \mathcal{B}) \cong_D (\mathcal{V}', \mathcal{B}')$   
*<proof>*

**end**

**context** *design-isomorphism*  
**begin**

### 10.3.2 Design Properties/Operations under Isomorphism

**lemma** *design-iso-point-rep-num-eq*:  
**assumes**  $p \in \mathcal{V}$   
**shows**  $\mathcal{B} \text{ rep } p = \mathcal{B}' \text{ rep } (\pi \text{ ` } p)$   
*<proof>*

**lemma** *design-iso-rep-numbers-eq*:  $\text{source.replication-numbers} = \text{target.replication-numbers}$   
*<proof>*

**lemma** *design-iso-block-size-eq*:  $bl \in \# \mathcal{B} \implies \text{card } bl = \text{card } (\pi \text{ ` } bl)$   
*<proof>*

**lemma** *design-iso-block-sizes-eq*:  $\text{source.sys-block-sizes} = \text{target.sys-block-sizes}$   
*<proof>*

**lemma** *design-iso-points-index-eq*:  
**assumes**  $ps \subseteq \mathcal{V}$   
**shows**  $\mathcal{B} \text{ index } ps = \mathcal{B}' \text{ index } (\pi \text{ ` } ps)$   
*<proof>*

**lemma** *design-iso-points-indices-imp*:  
**assumes**  $x \in \text{source.point-indices } t$   
**shows**  $x \in \text{target.point-indices } t$   
*<proof>*

**lemma** *design-iso-points-indices-eq*:  $\text{source.point-indices } t = \text{target.point-indices } t$   
*<proof>*

**lemma** *design-iso-block-intersect-num-eq*:  
**assumes**  $b1 \in \# \mathcal{B}$   
**assumes**  $b2 \in \# \mathcal{B}$   
**shows**  $b1 \mid \cap \mid b2 = (\pi \text{ ` } b1) \mid \cap \mid (\pi \text{ ` } b2)$   
*<proof>*

**lemma** *design-iso-inter-numbers-imp*:  
**assumes**  $x \in \text{source.intersection-numbers}$   
**shows**  $x \in \text{target.intersection-numbers}$   
*<proof>*

**lemma** *design-iso-intersection-numbers*:  $\text{source.intersection-numbers} = \text{target.intersection-numbers}$

*<proof>*

**lemma** *design-iso-n-intersect-num:*

**assumes**  $b1 \in \# \mathcal{B}$

**assumes**  $b2 \in \# \mathcal{B}$

**shows**  $b1 \mid \cap \mid_n b2 = ((\pi \cdot b1) \mid \cap \mid_n (\pi \cdot b2))$

*<proof>*

**lemma** *subdesign-iso-implies:*

**assumes** *sub-set-system*  $V B \mathcal{V} \mathcal{B}$

**shows** *sub-set-system*  $(\pi \cdot V) (\text{blocks-image } B \pi) \mathcal{V}' \mathcal{B}'$

*<proof>*

**lemma** *subdesign-image-is-design:*

**assumes** *sub-set-system*  $V B \mathcal{V} \mathcal{B}$

**assumes** *design*  $V B$

**shows** *design*  $(\pi \cdot V) (\text{blocks-image } B \pi)$

*<proof>*

**lemma** *sub-design-isomorphism:*

**assumes** *sub-set-system*  $V B \mathcal{V} \mathcal{B}$

**assumes** *design*  $V B$

**shows** *design-isomorphism*  $V B (\pi \cdot V) (\text{blocks-image } B \pi) \pi$

*<proof>*

**end**

**end**

**theory** *Design-Theory-Root*

**imports**

*Multisets-Extras*

*Design-Basics*

*Design-Operations*

*Block-Designs*

*BIBD*

*Resolvable-Designs*

*Group-Divisible-Designs*

*Designs-And-Graphs*

*Design-Isomorphisms*

*Sub-Designs*

**begin**

**end**

## References

- [1] E. A. Bender. Partitions of multisets. *Discrete Mathematics*, 9(4):301–311, Oct. 1974.
- [2] C. Berge. *Hypergraphs: Combinatorics of Finite Sets*. Number v. 45 in North-Holland Mathematical Library. North Holland : Distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York, 1989.
- [3] C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs / Edited by Charles J. Colbourn, Jeffrey H. Dinitz*. Chapman & Hall/CRC, 2nd edition, 2007.
- [4] S. Herke. Math3301 lecture notes in combinatorial design theory, July 2016.
- [5] L. H. Soicher. Designs, Groups and Computing. In A. Detinko, D. Flannery, and E. O’Brien, editors, *Probabilistic Group Theory, Combinatorics, and Computing*, Lecture Notes in Mathematics 2070, pages 83–107. Springer, 2013.
- [6] D. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer, 2004.