

# Fisher's Inequality: Linear Algebraic Proof Techniques for Combinatorics

Chelsea Edmonds and Lawrence C. Paulson

March 17, 2025

## Abstract

Linear algebraic techniques are powerful, yet often underrated tools in combinatorial proofs. This formalisation provides a library including matrix representations of incidence set systems, general formal proof techniques for the rank argument and linear bound argument, and finally a formalisation of a number of variations of the well-known Fisher's inequality. We build on our prior work formalising combinatorial design theory using a locale-centric approach, including extensions such as constant intersect designs and dual incidence systems. In addition to Fisher's inequality, we also formalise proofs on other incidence system properties using the incidence matrix representation, such as design existence, dual system relationships and incidence system isomorphisms. This formalisation is presented in the paper "Formalising Fisher's Inequality: Formal Linear Algebraic Techniques in Combinatorics", accepted to ITP 2022.

## Contents

<b>1</b>	<b>Miscellaneous Multset/Set Extras</b>	<b>3</b>
1.1	Set extras . . . . .	3
1.2	Multiset Extras . . . . .	3
1.3	Permutation on Sets and Multisets . . . . .	4
1.4	Lists . . . . .	4
1.5	Summation Rules . . . . .	5
<b>2</b>	<b>Matrix and Vector Additions</b>	<b>6</b>
2.1	Vector Extras . . . . .	6
2.2	Matrix Extras . . . . .	10
2.3	Vector and Matrix Homomorphism . . . . .	12
2.4	Zero One injections and homomorphisms . . . . .	13
<b>3</b>	<b>Miscellaneous Design Extras</b>	<b>16</b>
3.1	Extensions to existing Locales and Properties . . . . .	16
3.2	New Design Locales . . . . .	20

<b>4 Incidence Vectors and Matrices</b>	<b>22</b>
4.1 Incidence Vectors . . . . .	22
4.2 Incidence Matrices . . . . .	23
4.3 0-1 Matrices . . . . .	27
4.4 Ordered Incidence Systems . . . . .	32
4.5 Incidence Matrices on Design Subtypes . . . . .	41
4.6 Zero One Matrix Incidence System Existence . . . . .	45
4.7 Isomorphisms and Incidence Matrices . . . . .	48
<b>5 Dual Systems</b>	<b>48</b>
5.1 Dual Blocks . . . . .	49
5.2 Basic Dual Properties . . . . .	49
5.3 Incidence System Dual Properties . . . . .	50
5.4 Dual Properties for Design sub types . . . . .	51
5.5 Generalise Dual Concept . . . . .	54
<b>6 Rank Argument - General</b>	<b>55</b>
6.1 Row/Column Operations . . . . .	55
6.2 Rank and Linear Independence . . . . .	59
<b>7 Linear Bound Argument - General</b>	<b>60</b>
7.1 Vec Space Extensions . . . . .	60
7.2 Linear Bound Argument Lemmas . . . . .	61
<b>8 Fisher's Inequality</b>	<b>62</b>
8.1 Uniform Fisher's Inequality . . . . .	62
8.2 Generalised Fisher's Inequality . . . . .	64
<b>9 Matrices/Vectors mod x</b>	<b>65</b>
9.1 Basic Mod Context . . . . .	66
9.2 Mod Type . . . . .	68
9.3 Mat mod type . . . . .	70
<b>10 Variations on Fisher's Inequality</b>	<b>74</b>
10.1 Matrix mod properties . . . . .	74
10.2 The Odd-town Problem . . . . .	75

**Acknowledgements** The authors were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

# 1 Micellaneous Multset/Set Extras

```
theory Set-Multiset-Extras imports Design-Theory.Multisets-Extras HOL-Combinatorics.Multiset-Permutation
```

```
begin
```

## 1.1 Set extras

Minor set extras on cardinality and filtering

```
lemma equal-card-inter-fin-eq-sets: finite A ==> finite B ==> card A = card B ==>
```

```
card (A ∩ B) = card A ==> A = B  
(proof)
```

```
lemma insert-filter-set-true: P x ==> {a ∈ (insert x A) . P a} = insert x {a ∈ A . P a}  
(proof)
```

```
lemma insert-filter-set-false: ¬ P x ==> {a ∈ (insert x A) . P a} = {a ∈ A . P a}  
(proof)
```

## 1.2 Multiset Extras

Minor multiset extras on size and element reasoning

```
lemma obtain-two-items-mset:  
assumes size A > 1  
obtains x y where x ∈# A and y ∈# A - {#x#}  
(proof)
```

```
lemma obtain-two-items-mset-filter:  
assumes size {#a ∈# A . P a #} > 1  
obtains x y where x ∈# A and y ∈# A - {#x#} and P x and P y  
(proof)
```

```
lemma size-count-mset-ss:  
assumes finite B  
assumes (set-mset A) ⊆ B  
shows size A = (∑ x ∈ B . count A x)  
(proof)
```

```
lemma mset-list-by-index: mset (xs) = image-mset (λ i . (xs ! i)) (mset-set {..  
xs})  
(proof)
```

```
lemma count-mset-split-image-filter:  
assumes ∀ x. x ∈# A ==> a ≠ g x  
shows count (image-mset (λx. if P x then a else g x) A) a = size (filter-mset P A)  
(proof)
```

```

lemma count-mset-split-image-filter-not:
  assumes  $\bigwedge x. x \in \#A \implies b \neq f x$ 
  shows count (image-mset ( $\lambda x. \text{if } P x \text{ then } f x \text{ else } b$ ) A) b = size (filter-mset ( $\lambda x. \neg P x$ ) A)
   $\langle \text{proof} \rangle$ 

lemma removeAll-size-lt: size (removeAll-mset C M)  $\leq$  size M
   $\langle \text{proof} \rangle$ 

lemma mset-image-eq-filter-eq:  $A = \text{image-mset } f B \implies$ 
  filter-mset P A = (image-mset f (filter-mset ( $\lambda x. P(f x)$ ) B))
   $\langle \text{proof} \rangle$ 

```

### 1.3 Permutation on Sets and Multisets

```

lemma elem-permutation-of-set-empty-iff: finite A  $\implies$  xs  $\in$  permutations-of-set
A  $\implies$ 
  xs = []  $\longleftrightarrow$  A = {}
   $\langle \text{proof} \rangle$ 

lemma elem-permutation-of-mset-empty-iff: xs  $\in$  permutations-of-multiset A  $\implies$ 
  xs = []  $\longleftrightarrow$  A = {#}
   $\langle \text{proof} \rangle$ 

```

### 1.4 Lists

Further lemmas on the relationship between lists and multisets

```

lemma count-distinct-mset-list-index: i1 < length xs  $\implies$  i2 < length xs  $\implies$  i1  $\neq$ 
i2  $\implies$ 
  distinct-mset (mset xs)  $\implies$  xs ! i1  $\neq$  xs ! i2
   $\langle \text{proof} \rangle$ 

```

```

lemma index-remove1-mset-ne:
  assumes x  $\in \# (\text{mset } xs)$ 
  assumes y  $\in \# \text{remove1-mset } x (\text{mset } xs)$ 
  assumes xs ! j1 = x
  assumes j1 < length xs
  obtains j2 where xs ! j2 = y and j2 < length xs and j1  $\neq$  j2
   $\langle \text{proof} \rangle$ 

```

```

lemma count-list-mset: count-list xs x = count (mset xs) x
   $\langle \text{proof} \rangle$ 

```

```

lemma count-min-2-indices-lt:
  assumes i1 < i2
  assumes xs ! i1 = x
  assumes xs ! i2 = x
  assumes i1 < length xs i2 < length xs
  shows count (mset xs) x  $\geq$  2

```

$\langle proof \rangle$

**lemma** count-min-2-indices:  $i1 \neq i2 \implies xs ! i1 = x \implies xs ! i2 = x \implies i1 < length xs \implies i2 < length xs \implies count(mset xs) x \geq 2$   
 $\langle proof \rangle$

**lemma** obtain-set-list-item:  
  **assumes**  $x \in set xs$   
  **obtains**  $i$  **where**  $i < length xs$  **and**  $xs ! i = x$   
 $\langle proof \rangle$

## 1.5 Summation Rules

Some lemmas to make it simpler to work with double and triple summations

**context** comm-monoid-add  
**begin**

**lemma** sum-reorder-triple:  $(\sum l \in A . (\sum i \in B . (\sum j \in C . g l i j))) = (\sum i \in B . (\sum j \in C . (\sum l \in A . g l i j)))$   
 $\langle proof \rangle$

**lemma** double-sum-mult-hom:  
  **fixes**  $k :: 'b :: \{comm-ring-1\}$   
  **shows**  $(\sum i \in A . (\sum j \in g i . k * (f i j))) = k * (\sum i \in A . (\sum j \in g i . f i j))$   
 $\langle proof \rangle$

**lemma** double-sum-split-case:  
  **assumes** finite  $A$   
  **shows**  $(\sum i \in A . (\sum j \in A . f i j)) = (\sum i \in A . (f i i)) + (\sum i \in A . (\sum j \in (A - \{i\}) . f i j))$   
 $\langle proof \rangle$

**lemma** double-sum-split-case2:  $(\sum i \in A . (\sum j \in A . g i j)) = (\sum i \in A . (g i i)) + (\sum i \in A . (\sum j \in \{a \in A . a \neq i\} . g i j))$   
 $\langle proof \rangle$

**end**

**context** comm-ring-1  
**begin**

**lemma** double-sum-split-case-square:  
  **assumes** finite  $A$   
  **shows**  $(\sum i \in A . f i)^2 = (\sum i \in A . (f i * f i)) + (\sum i \in A . (\sum j \in (A - \{i\}) . f i * f j))$   
 $\langle proof \rangle$

**lemma** double-sum-split-square-diff: finite  $\{0..\}$   $\implies$

$$\begin{aligned} & (\sum i \in \{0..<x\} . (\sum j \in (\{0..<x\} - \{i\}) . c i * c j)) = \\ & (\sum i \in \{0..<x\} . c i) \hat{\wedge} 2 - (\sum i \in \{0..<x\} . c i * c i) \end{aligned}$$

**end**  
**end**

## 2 Matrix and Vector Additions

**theory** *Matrix-Vector-Extras imports Set-Multiset-Extras Jordan-Normal-Form.Matrix*

*Design-Theory.Multisets-Extras Groebner-Bases.Macaulay-Matrix Polynomial-Factorization.Missing-List begin*

### 2.1 Vector Extras

For ease of use, a number of additions to the existing vector library as initially developed in the JNF AFP Entry, are given below

We define the concept of summing up elements of a vector

**definition (in comm-monoid-add)** *sum-vec* :: *'a vec*  $\Rightarrow$  *'a* **where**  
*sum-vec v*  $\equiv$  *sum* ( $\lambda i . v \$ i$ ) {0..<*dim-vec v*}

**lemma** *sum-vec-vNil*[simp]: *sum-vec vNil* = 0  
 $\langle proof \rangle$

**lemma** *sum-vec-vCons*: *sum-vec (vCons a v)* = *a + sum-vec v*  
 $\langle proof \rangle$

**lemma** *sum-vec-list*: *sum-list (list-of-vec v)* = *sum-vec v*  
 $\langle proof \rangle$

**lemma** *sum-vec-mset*: *sum-vec v* = ( $\sum x \in \# (mset (list-of-vec v)) . x$ )  
 $\langle proof \rangle$

**lemma** *dim-vec-vCons-ne-0*: *dim-vec (vCons a v)* > 0  
 $\langle proof \rangle$

**lemma** *sum-vec-vCons-lt*:  
**assumes**  $\bigwedge i . i < dim-vec (vCons a v) \implies (vCons a v) \$ i \leq (n :: int)$   
**assumes** *sum-vec v*  $\leq m$   
**shows** *sum-vec (vCons a v)*  $\leq n + m$   
 $\langle proof \rangle$

**lemma** *sum-vec-one-zero*:  
**assumes**  $\bigwedge i . i < dim-vec (v :: int vec) \implies v \$ i \leq (1 :: int)$   
**shows** *sum-vec v*  $\leq dim-vec v$   
 $\langle proof \rangle$

Definition to convert a vector to a multiset

```

definition vec-mset:: 'a vec  $\Rightarrow$  'a multiset where
vec-mset v  $\equiv$  image-mset (vec-index v) (mset-set {.. $<$ dim-vec v})

lemma vec-elem-exists-mset: ( $\exists$  i  $\in$  {.. $<$ dim-vec v}. v $ i = x)  $\longleftrightarrow$  x  $\in$  vec-mset
v
    {proof}

lemma mset-vec-same-size: dim-vec v = size (vec-mset v)
{proof}

lemma mset-vec-eq-mset-list: vec-mset v = mset (list-of-vec v)
{proof}

lemma vec-mset-img-map: image-mset f (mset (xs)) = vec-mset (map-vec f (vec-of-list
xs))
{proof}

lemma vec-mset-vNil: vec-mset vNil = {#}
{proof}

lemma vec-mset-vCons: vec-mset (vCons x v) = add-mset x (vec-mset v)
{proof}

lemma vec-mset-set: vec-set v = set-mset (vec-mset v)
{proof}

lemma vCons-set-contains-in: a  $\in$  setv v  $\implies$  setv (vCons a v) = setv v
{proof}

lemma vCons-set-contains-add: a  $\notin$  setv v  $\implies$  setv (vCons a v) = setv v  $\cup$  {a}
{proof}

lemma setv-vec-mset-not-in-iff: a  $\notin$  setv v  $\longleftrightarrow$  a  $\notin$  vec-mset v
{proof}
```

Abbreviation for counting occurrences of an element in a vector

```

abbreviation count-vec v a  $\equiv$  count (vec-mset v) a

lemma vec-count-lt-dim: count-vec v a  $\leq$  dim-vec v
{proof}

lemma count-vec-empty: dim-vec v = 0  $\implies$  count-vec v a = 0
{proof}

lemma count-vec-vNil: count-vec vNil a = 0
{proof}

lemma count-vec-vCons: count-vec (vCons aa v) a = (if (aa = a) then count-vec
```

$v \ a + 1 \text{ else } \text{count-vec } v \ a)$   
 $\langle \text{proof} \rangle$

**lemma** elem-exists-count-min:  $\exists i \in \{\dots < \text{dim-vec } v\}. v \$ i = x \implies \text{count-vec } v x \geq 1$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-count-mset:  $\text{vec-mset } v = \text{image-mset } f A \implies \text{count-vec } v a = \text{count } (\text{image-mset } f A) a$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-alt-list:  $\text{count-vec } v a = \text{length } (\text{filter } (\lambda y. a = y) (\text{list-of-vec } v))$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-alt:  $\text{count-vec } v x = \text{card } \{ i. v \$ i = x \wedge i < \text{dim-vec } v \}$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-sum-ones:  
**fixes**  $v :: 'a :: \{\text{ring-1}\} \text{ vec}$   
**assumes**  $\bigwedge i. i < \text{dim-vec } v \implies v \$ i = 1 \vee v \$ i = 0$   
**shows** of-nat (count-vec  $v 1$ ) = sum-vec  $v$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-two-elems:  
**fixes**  $v :: 'a :: \{\text{zero-neg-one}\} \text{ vec}$   
**assumes**  $\bigwedge i. i < \text{dim-vec } v \implies v \$ i = 1 \vee v \$ i = 0$   
**shows** count-vec  $v 1 + \text{count-vec } v 0 = \text{dim-vec } v$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-sum-zeros:  
**fixes**  $v :: 'a :: \{\text{ring-1}\} \text{ vec}$   
**assumes**  $\bigwedge i. i < \text{dim-vec } v \implies v \$ i = 1 \vee v \$ i = 0$   
**shows** of-nat (count-vec  $v 0$ ) = of-nat (dim-vec  $v$ ) - sum-vec  $v$   
 $\langle \text{proof} \rangle$

**lemma** count-vec-sum-ones-alt:  
**fixes**  $v :: 'a :: \{\text{ring-1}\} \text{ vec}$   
**assumes**  $\text{vec-set } v \subseteq \{0, 1\}$   
**shows** of-nat (count-vec  $v 1$ ) = sum-vec  $v$   
 $\langle \text{proof} \rangle$

**lemma** setv-not-in-count0-iff:  $a \notin \text{set}_v v \longleftrightarrow \text{count-vec } v a = 0$   
 $\langle \text{proof} \rangle$

**lemma** sum-count-vec:  
**assumes** finite ( $\text{set}_v v$ )  
**shows**  $(\sum i \in \text{set}_v v. \text{count-vec } v i) = \text{dim-vec } v$   
 $\langle \text{proof} \rangle$

```

lemma sum-setv-subset-eq:
  assumes finite A
  assumes setv v ⊆ A
  shows (∑ i ∈ setv v. count-vec v i) = (∑ i ∈ A. count-vec v i)
  ⟨proof⟩

lemma sum-count-vec-subset: finite A ⇒ setv v ⊆ A ⇒ (∑ i ∈ A. count-vec v i) = dim-vec v
  ⟨proof⟩

  An abbreviation for checking if an element is in a vector

abbreviation vec-contains :: 'a ⇒ 'a vec ⇒ bool (infix ∈\$ 50)where
  a ∈\$ v ≡ a ∈ setv v

lemma vec-set-mset-contains-iff: a ∈\$ v ⇐⇒ a ∈# vec-mset v
  ⟨proof⟩

lemma vec-contains-count-gt1-iff: a ∈\$ v ⇐⇒ count-vec v a ≥ 1
  ⟨proof⟩

lemma vec-contains-obtains-index:
  assumes a ∈\$ v
  obtains i where i < dim-vec v and v \$ i = a
  ⟨proof⟩

lemma vec-count-eq-list-count: count (mset xs) a = count-vec (vec-of-list xs) a
  ⟨proof⟩

lemma vec-contains-col-elements-mat:
  assumes j < dim-col M
  assumes a ∈\$ col M j
  shows a ∈ elements-mat M
  ⟨proof⟩

lemma vec-contains-row-elements-mat:
  assumes i < dim-row M
  assumes a ∈\$ row M i
  shows a ∈ elements-mat M
  ⟨proof⟩

lemma vec-contains-img: a ∈\$ v ⇒ f a ∈\$ (map-vec f v)
  ⟨proof⟩

```

The existing vector library contains the identity and zero vectors, but no definition of a vector where all elements are 1, as defined below

```

definition all-ones-vec :: nat ⇒ 'a :: {zero,one} vec (⟨uv⟩) where
  uv n ≡ vec n (λ i. 1)

```

```

lemma dim-vec-all-ones[simp]: dim-vec (uv n) = n

```

$\langle proof \rangle$

**lemma** all-ones-index [simp]:  $i < n \implies u_v n \$ i = 1$   
 $\langle proof \rangle$

**lemma** dim-vec-mult-vec-mat [simp]:  $\text{dim-vec } (v \cdot_v A) = \text{dim-col } A$   
 $\langle proof \rangle$

**lemma** all-ones-vec-smult[simp]:  $i < n \implies ((k :: ('a :: \{one, zero, monoid-mult\})) \cdot_v (u_v n)) \$ i = k$   
 $\langle proof \rangle$

Extra lemmas on existing vector operations

**lemma** smult-scalar-prod-sum:  
**fixes**  $x :: 'a :: \{\text{comm-ring-1}\}$   
**assumes**  $vx \in \text{carrier-vec } n$   
**assumes**  $vy \in \text{carrier-vec } n$   
**shows**  $(\sum_{i \in \{0..n\}} ((x \cdot_v vx) \$ i) * ((y \cdot_v vy) \$ i)) = x * y * (vx \cdot vy)$   
 $\langle proof \rangle$

**lemma** scalar-prod-double-sum-fn-vec:  
**fixes**  $c :: nat \Rightarrow ('a :: \{\text{comm-semiring-0}\})$   
**fixes**  $f :: nat \Rightarrow 'a \text{ vec}$   
**assumes**  $\bigwedge j . j < k \implies \text{dim-vec } (f j) = n$   
**shows**  $(\text{vec } n (\lambda i . \sum_{j=0..k} c j * (f j) \$ i)) \cdot (\text{vec } n (\lambda i . \sum_{j=0..k} c j * (f j) \$ i)) =$   
 $(\sum_{j1 \in \{0..k\}} c j1 * c j1 * ((f j1) \cdot (f j1))) +$   
 $(\sum_{j1 \in \{0..k\}} (\sum_{j2 \in \{0..k\} - \{j1\}} c j1 * c j2 * ((f j1) \cdot (f j2))))$   
 $\langle proof \rangle$

**lemma** vec-prod-zero:  $(0_v n) \cdot (0_v n) = 0$   
 $\langle proof \rangle$

**lemma** map-vec-compose:  $\text{map-vec } f (\text{map-vec } g v) = \text{map-vec } (f \circ g) v$   
 $\langle proof \rangle$

## 2.2 Matrix Extras

As with vectors, the all ones mat definition defines the concept of a matrix where all elements are 1

**definition** all-ones-mat ::  $nat \Rightarrow 'a :: \{\text{zero, one}\} \text{ mat } (\langle J_m \rangle)$  **where**  
 $J_m n \equiv \text{mat } n n (\lambda (i,j). 1)$

**lemma** all-ones-mat-index[simp]:  $i < \text{dim-row } (J_m n) \implies j < \text{dim-col } (J_m n)$   
 $\implies J_m n \$\$ (i, j) = 1$   
 $\langle proof \rangle$

**lemma** all-ones-mat-dim-row[simp]:  $\text{dim-row } (J_m n) = n$   
 $\langle proof \rangle$

**lemma** *all-ones-mat-dim-col*[simp]:  $\text{dim-col} (J_m n) = n$   
 $\langle \text{proof} \rangle$

Basic lemmas on existing matrix operations

**lemma** *index-mult-vec-mat*[simp]:  $j < \text{dim-col } A \implies (v \ v * A) \$ j = v \cdot \text{col } A j$   
 $\langle \text{proof} \rangle$

**lemma** *transpose-mat-mult-entries*:  $i < \text{dim-row } A \implies j < \text{dim-row } A \implies$   
 $(A * A^T) \$\$ (i, j) = (\sum k \in \{0..<(\text{dim-col } A)\}. (A \$\$ (i, k)) * (A \$\$ (j, k)))$   
 $\langle \text{proof} \rangle$

**lemma** *transpose-mat-elems*: *elements-mat*  $A = \text{elements-mat } A^T$   
 $\langle \text{proof} \rangle$

**lemma** *row-elems-subset-mat*:  $i < \text{dim-row } N \implies \text{vec-set} (\text{row } N i) \subseteq \text{elements-mat } N$   
 $\langle \text{proof} \rangle$

**lemma** *col-elems-subset-mat*:  $i < \text{dim-col } N \implies \text{vec-set} (\text{col } N i) \subseteq \text{elements-mat } N$   
 $\langle \text{proof} \rangle$

**lemma** *obtain-row-index*:  
**assumes**  $r \in \text{set} (\text{rows } M)$   
**obtains**  $i$  **where**  $\text{row } M i = r$  **and**  $i < \text{dim-row } M$   
 $\langle \text{proof} \rangle$

**lemma** *row-prop-cond*:  $(\bigwedge i. i < \text{dim-row } M \implies P (\text{row } M i)) \implies r \in \text{set} (\text{rows } M) \implies P r$   
 $\langle \text{proof} \rangle$

**lemma** *obtain-col-index*:  
**assumes**  $c \in \text{set} (\text{cols } M)$   
**obtains**  $j$  **where**  $\text{col } M j = c$  **and**  $j < \text{dim-col } M$   
 $\langle \text{proof} \rangle$

**lemma** *col-prop-cond*:  $(\bigwedge j. j < \text{dim-col } M \implies P (\text{col } M j)) \implies c \in \text{set} (\text{cols } M) \implies P c$   
 $\langle \text{proof} \rangle$

Lemmas on the *map-mat* definition

**lemma** *row-map-mat*[simp]:  
**assumes**  $i < \text{dim-row } A$  **shows**  $\text{row} (\text{map-mat } f A) i = \text{map-vec } f (\text{row } A i)$   
 $\langle \text{proof} \rangle$

**lemma** *map-vec-mat-rows*:  $\text{map} (\text{map-vec } f) (\text{rows } M) = \text{rows} ((\text{map-mat } f) M)$   
 $\langle \text{proof} \rangle$

**lemma** *map-vec-mat-cols*:  $\text{map}(\text{map-vec } f)(\text{cols } M) = \text{cols}((\text{map-mat } f) M)$   
 $\langle \text{proof} \rangle$

**lemma** *map-mat-compose*:  $\text{map-mat } f (\text{map-mat } g A) = \text{map-mat}(f \circ g) A$   
 $\langle \text{proof} \rangle$

**lemmas** *map-mat-elements* = *elements-mat-map*

Reasoning on sets and multisets of matrix elements

**lemma** *set-cols-carrier*:  $A \in \text{carrier-mat } m n \implies v \in \text{set}(\text{cols } A) \implies v \in \text{carrier-vec } m$   
 $\langle \text{proof} \rangle$

**lemma** *mset-cols-index-map*:  $\text{image-mset}(\lambda j. \text{col } M j) (\text{mset-set } \{0..< \text{dim-col } M\}) = \text{mset}(\text{cols } M)$   
 $\langle \text{proof} \rangle$

**lemma** *mset-rows-index-map*:  $\text{image-mset}(\lambda i. \text{row } M i) (\text{mset-set } \{0..< \text{dim-row } M\}) = \text{mset}(\text{rows } M)$   
 $\langle \text{proof} \rangle$

**lemma** *index-to-col-card-size-prop*:  
**assumes**  $i < \text{dim-row } M$   
**assumes**  $\bigwedge j. j < \text{dim-col } M \implies P j \longleftrightarrow Q (\text{col } M j)$   
**shows**  $\text{card}\{j . j < \text{dim-col } M \wedge P j\} = \text{size}\{\#c \in \#(\text{mset}(\text{cols } M)) . Q c \#\}$   
 $\langle \text{proof} \rangle$

**lemma** *index-to-row-card-size-prop*:  
**assumes**  $j < \text{dim-col } M$   
**assumes**  $\bigwedge i. i < \text{dim-row } M \implies P i \longleftrightarrow Q (\text{row } M i)$   
**shows**  $\text{card}\{i . i < \text{dim-row } M \wedge P i\} = \text{size}\{\#r \in \#(\text{mset}(\text{rows } M)) . Q r \#\}$   
 $\langle \text{proof} \rangle$

**lemma** *setv-row-subset-mat-elems*:  
**assumes**  $v \in \text{set}(\text{rows } M)$   
**shows**  $\text{set}_v v \subseteq \text{elements-mat } M$   
 $\langle \text{proof} \rangle$

**lemma** *setv-col-subset-mat-elems*:  
**assumes**  $v \in \text{set}(\text{cols } M)$   
**shows**  $\text{set}_v v \subseteq \text{elements-mat } M$   
 $\langle \text{proof} \rangle$

## 2.3 Vector and Matrix Homomorphism

We extend on the existing lemmas on homomorphism mappings as applied to vectors and matrices

**context** *semiring-hom*

```

begin

lemma vec-hom-smult2:
assumes dim-vec v2 ≤ dim-vec v1
shows hom (v1 · v2) = vech v1 · vech v2
⟨proof⟩

end

lemma map-vec-vCons: vCons (f a) (map-vec f v) = map-vec f (vCons a v)
⟨proof⟩

context inj-zero-hom
begin

lemma vec-hom-zero-iff[simp]: (map-vec hom x = 0v n) = (x = 0v n)
⟨proof⟩

lemma mat-hom-inj: map-mat hom A = map-mat hom B ⇒ A = B
⟨proof⟩

lemma vec-hom-inj: map-vec hom v = map-vec hom w ⇒ v = w
⟨proof⟩

lemma vec-hom-set-distinct-iff:
fixes xs :: 'a vec list
shows distinct xs ↔ distinct (map (map-vec hom) xs)
⟨proof⟩

lemma vec-hom-mset: image-mset hom (vec-mset v) = vec-mset (map-vec hom v)
⟨proof⟩

lemma vec-hom-set: hom ` setv v = setv (map-vec hom v)
⟨proof⟩

end

```

## 2.4 Zero One injections and homomorphisms

Define a locale to encapsulate when a function is injective on a certain set (i.e. not a universal homomorphism for the type

```

locale injective-lim =
fixes A :: 'a set
fixes f :: 'a ⇒ 'b assumes injectivity-lim: ∀x y. x ∈ A ⇒ y ∈ A ⇒ f x = f
y ⇒ x = y
begin
lemma eq-iff[simp]: x ∈ A ⇒ y ∈ A ⇒ f x = f y ↔ x = y ⟨proof⟩
lemma inj-on-f: inj-on f A ⟨proof⟩

```

```

end

sublocale injective ⊆ injective-lim Univ
  ⟨proof⟩

context injective-lim
begin

lemma mat-hom-inj-lim:
  assumes elements-mat M ⊆ A and elements-mat N ⊆ A
  shows map-mat f M = map-mat f N ==> M = N
  ⟨proof⟩

lemma vec-hom-inj-lim: assumes setv v ⊆ A setw w ⊆ A
  shows map-vec f v = map-vec f w ==> v = w
  ⟨proof⟩

lemma lim-inj-hom-count-vec:
  assumes setv v ⊆ A
  assumes x ∈ A
  shows count-vec v x = count-vec (map-vec f v) (f x)
  ⟨proof⟩

lemma vec-hom-lim-set-distinct-iff:
  fixes xs :: 'a vec list
  assumes ⋀ v . v ∈ set (xs) ==> setv v ⊆ A
  shows distinct xs <=> distinct (map (map-vec f) xs)
  ⟨proof⟩

lemma mat-rows-hom-lim-distinct-iff:
  assumes elements-mat M ⊆ A
  shows distinct (rows M) <=> distinct (map (map-vec f) (rows M))
  ⟨proof⟩

lemma mat-cols-hom-lim-distinct-iff:
  assumes elements-mat M ⊆ A
  shows distinct (cols M) <=> distinct (map (map-vec f) (cols M))
  ⟨proof⟩

end

locale inj-on-01-hom = zero-hom + one-hom + injective-lim {0, 1} hom
begin

lemma inj-0-iff: x ∈ {0, 1} ==> hom x = 0 <=> x = 0
  ⟨proof⟩

lemma inj-1-iff: x ∈ {0, 1} ==> hom x = 1 <=> x = 1

```

```

⟨proof⟩

end

context zero-neq-one
begin

definition of-zero-neq-one :: 'b :: {zero-neq-one} ⇒ 'a where
of-zero-neq-one x ≡ if (x = 0) then 0 else 1

lemma of-zero-neq-one-1 [simp]: of-zero-neq-one 1 = 1
⟨proof⟩

lemma of-zero-neq-one-0 [simp]: of-zero-neq-one 0 = 0
⟨proof⟩

lemma of-zero-neq-one-0-iff[iff]: of-zero-neq-one x = 0 ↔ x = 0
⟨proof⟩

lemma of-zero-neq-one-lim-eq: x ∈ {0, 1} ⇒ y ∈ {0, 1} ⇒ of-zero-neq-one x
= of-zero-neq-one y ↔ x = y
⟨proof⟩

end

interpretation of-zero-hom: zero-hom-0 of-zero-neq-one
⟨proof⟩

interpretation of-injective-lim: injective-lim {0, 1} of-zero-neq-one
⟨proof⟩

interpretation of-inj-on-01-hom: inj-on-01-hom of-zero-neq-one
⟨proof⟩

We want the ability to transform any 0-1 vector or matrix to another 'c' type

definition lift-01-vec :: 'b :: {zero-neq-one} vec ⇒ 'c :: {zero-neq-one} vec where
lift-01-vec v ≡ map-vec of-zero-neq-one v

lemma lift-01-vec-simp[simp]: dim-vec (lift-01-vec v) = dim-vec v
i < dim-vec v ⇒ (lift-01-vec v) $ i = of-zero-neq-one (v $ i)
⟨proof⟩

lemma lift-01-vec-count:
assumes setv v ⊆ {0, 1}
assumes x ∈ {0, 1}
shows count-vec v x = count-vec (lift-01-vec v) (of-zero-neq-one x)

```

```

⟨proof⟩

definition lift-01-mat :: 'b :: {zero-neq-one} mat ⇒ 'c :: {zero-neq-one} mat
where
lift-01-mat M ≡ map-mat of-zero-neq-one M

lemma lift-01-mat-simp[simp]: dim-row (lift-01-mat M) = dim-row M
dim-col (lift-01-mat M) = dim-col M
i < dim-row M ⇒ j < dim-col M ⇒ (lift-01-mat M) $$ (i, j) = of-zero-neq-one
(M $$ (i, j))
⟨proof⟩

lemma lift-01-mat-carrier: lift-01-mat M ∈ carrier-mat (dim-row M) (dim-col M)
⟨proof⟩

end

```

### 3 Micellaneous Design Extras

Extension's to the author's previous entry on Design Theory

```

theory Design-Extras imports Set-Multiset-Extras Design-Theory.BIBD
begin

```

#### 3.1 Extensions to existing Locales and Properties

Extend lemmas on intersection number

```

lemma inter-num-max-bound:
assumes finite b1 finite b2
shows b1 |∩| b2 ≤ card b1 b1 |∩| b2 ≤ card b2
⟨proof⟩

```

```

lemma inter-eq-blocks-eq-card: card b1 = card b2 ⇒ finite b1 ⇒ finite b2 ⇒
b1 |∩| b2 = card b1
⇒ b1 = b2
⟨proof⟩

```

```

lemma inter-num-of-eq-blocks: b1 = b2 ⇒ b1 |∩| b2 = card b1
⟨proof⟩

```

```

lemma intersect-num-same-eq-size[simp]: bl |∩| bl = card bl
⟨proof⟩

```

```

lemma index-lt-rep-general: x ∈ ps ⇒ B index ps ≤ B rep x
⟨proof⟩

```

```

context incidence-system
begin

```

```

lemma block-size-alt:
  assumes bl ∈#  $\mathcal{B}$ 
  shows card bl = card {x ∈  $\mathcal{V}$  . x ∈ bl}
  ⟨proof⟩

```

```

lemma complement-image:  $\mathcal{B}^C = \text{image-mset block-complement } \mathcal{B}$ 
  ⟨proof⟩

```

```

lemma point-in-block-rep-min-iff:
  assumes x ∈  $\mathcal{V}$ 
  shows ∃ bl . bl ∈#  $\mathcal{B}$  ∧ x ∈ bl  $\longleftrightarrow$  ( $\mathcal{B}$  rep x > 0)
  ⟨proof⟩

```

```

lemma points-inter-num-rep:
  assumes b1 ∈#  $\mathcal{B}$  and b2 ∈#  $\mathcal{B} - \{\#b1\}$ 
  shows card {v ∈  $\mathcal{V}$  . v ∈ b1 ∧ v ∈ b2} = b1 |∩| b2
  ⟨proof⟩

```

Extensions on design operation lemmas

```

lemma del-block-b:
  bl ∈#  $\mathcal{B} \implies \text{size (del-block bl)} = b - 1$ 
  bl ∉#  $\mathcal{B} \implies \text{size (del-block bl)} = b$ 
  ⟨proof⟩

```

```

lemma del-block-points-index:
  assumes ps ⊆  $\mathcal{V}$ 
  assumes card ps = 2
  assumes bl ∈#  $\mathcal{B}$ 
  shows ps ⊆ bl  $\implies$  points-index (del-block bl) ps = points-index  $\mathcal{B}$  ps - 1
   $\neg (ps \subseteq bl) \implies$  points-index (del-block bl) ps = points-index  $\mathcal{B}$  ps
  ⟨proof⟩

```

**end**

Extensions to properties of design sub types

```

context finite-incidence-system
begin

```

```

lemma complete-block-size-eq-points: bl ∈#  $\mathcal{B} \implies \text{card bl} = v \implies bl = \mathcal{V}$ 
  ⟨proof⟩

```

```

lemma complete-block-all-subsets: bl ∈#  $\mathcal{B} \implies \text{card bl} = v \implies ps \subseteq \mathcal{V} \implies ps \subseteq bl$ 
  ⟨proof⟩

```

```

lemma del-block-complete-points-index: ps ⊆  $\mathcal{V} \implies \text{card ps} = 2 \implies bl \in# \mathcal{B} \implies$ 
 $\text{card bl} = v \implies$ 
  points-index (del-block bl) ps = points-index  $\mathcal{B}$  ps - 1
  ⟨proof⟩

```

```

end

context design
begin

lemma block-num-rep-bound:  $b \leq (\sum_{x \in \mathcal{V}} \mathcal{B} \text{ rep } x)$ 
(proof)

end

context proper-design
begin

lemma del-block-proper:
  assumes  $b > 1$ 
  shows proper-design  $\mathcal{V}$  (del-block  $bl$ )
(proof)

end

context simple-design
begin

lemma inter-num-lt-block-size-strict:
  assumes  $bl1 \in \# \mathcal{B}$ 
  assumes  $bl2 \in \# \mathcal{B}$ 
  assumes  $bl1 \neq bl2$ 
  assumes  $\text{card } bl1 = \text{card } bl2$ 
  shows  $bl1 \cap bl2 < \text{card } bl1$   $bl1 \cap bl2 < \text{card } bl2$ 
(proof)

lemma block-mset-distinct: distinct-mset  $\mathcal{B}$  (proof)

end

context constant-rep-design
begin

lemma index-lt-const-rep:
  assumes  $ps \subseteq \mathcal{V}$ 
  assumes  $ps \neq \{\}$ 
  shows  $\mathcal{B} \text{ index } ps \leq r$ 
(proof)

end

context t-wise-balance
begin

```

```

lemma obtain-t-subset-with-point:
  assumes  $x \in \mathcal{V}$ 
  obtains  $ps$  where  $ps \subseteq \mathcal{V}$  and  $\text{card } ps = t$  and  $x \in ps$ 
   $\langle proof \rangle$ 

lemma const-index-lt-rep:
  assumes  $x \in \mathcal{V}$ 
  shows  $\Lambda_t \leq \mathcal{B} \text{ rep } x$ 
   $\langle proof \rangle$ 

end

context pairwise-balance
begin

lemma index-zero-iff:  $\Lambda = 0 \longleftrightarrow (\forall bl \in \# \mathcal{B} . \text{card } bl = 1)$ 
   $\langle proof \rangle$ 

lemma count-complete-lt-balance:  $\text{count } \mathcal{B} \mathcal{V} \leq \Lambda$ 
   $\langle proof \rangle$ 

lemma eq-index-rep-imp-complete:
  assumes  $\Lambda = \mathcal{B} \text{ rep } x$ 
  assumes  $x \in \mathcal{V}$ 
  assumes  $bl \in \# \mathcal{B}$ 
  assumes  $x \in bl$ 
  shows  $\text{card } bl = v$ 
   $\langle proof \rangle$ 

lemma incomplete-index-strict-lt-rep:
  assumes  $\bigwedge bl. bl \in \# \mathcal{B} \implies \text{incomplete-block } bl$ 
  assumes  $x \in \mathcal{V}$ 
  assumes  $\Lambda > 0$ 
  shows  $\Lambda < \mathcal{B} \text{ rep } x$ 
   $\langle proof \rangle$ 

```

Construct new PBD's from existing PBD's

```

lemma remove-complete-block-pbd:
  assumes  $b \geq 2$ 
  assumes  $bl \in \# \mathcal{B}$ 
  assumes  $\text{card } bl = v$ 
  shows pairwise-balance  $\mathcal{V} (\text{del-block } bl) (\Lambda - 1)$ 
   $\langle proof \rangle$ 

lemma remove-complete-block-pbd-alt:
  assumes  $b \geq 2$ 
  assumes  $bl \in \# \mathcal{B}$ 
  assumes  $bl = \mathcal{V}$ 

```

```

shows pairwise-balance  $\mathcal{V}$  (del-block bl) ( $\Lambda - 1$ )
⟨proof⟩

lemma b-gt-index:b  $\geq \Lambda$ 
⟨proof⟩

lemma remove-complete-blocks-set-pbd:
assumes  $x < \Lambda$ 
assumes size  $A = x$ 
assumes  $A \subset\# \mathcal{B}$ 
assumes  $\bigwedge a. a \in\# A \implies a = \mathcal{V}$ 
shows pairwise-balance  $\mathcal{V}$  ( $\mathcal{B} - A$ ) ( $\Lambda - x$ )
⟨proof⟩

lemma remove-all-complete-blocks-pbd:
assumes count  $\mathcal{B}$   $\mathcal{V} < \Lambda$ 
shows pairwise-balance  $\mathcal{V}$  (removeAll-mset  $\mathcal{V}$   $\mathcal{B}$ ) ( $\Lambda - (\text{count } \mathcal{B} \mathcal{V})$ ) (is pairwise-balance  $\mathcal{V}$  ? $\mathcal{B}$  ? $\Lambda$ )
⟨proof⟩

end

context bibd
begin
lemma symmetric-bibdIII:  $r = k \implies$  symmetric-bibd  $\mathcal{V}$   $\mathcal{B}$   $k$   $\Lambda$ 
⟨proof⟩
end

```

### 3.2 New Design Locales

We establish a number of new locales and link them to the existing locale hierarchy in order to reason in contexts requiring specific combinations of contexts

Regular t-wise balance

```

locale regular-t-wise-balance = t-wise-balance + constant-rep-design
begin

lemma reg-index-lt-rep:
shows  $\Lambda_t \leq r$ 
⟨proof⟩

end

locale regular-pairwise-balance = regular-t-wise-balance  $\mathcal{V}$   $\mathcal{B}$  ? $\Lambda$  r + pairwise-balance
 $\mathcal{V}$   $\mathcal{B}$   $\Lambda$ 
for  $\mathcal{V}$  and  $\mathcal{B}$  and  $\Lambda$  and r

```

Const Intersect Design

This is the dual of a balanced design, and used extensively in the remaining formalisation

```

locale const-intersect-design = proper-design +
  fixes m :: nat
  assumes const-intersect:  $b1 \in \# \mathcal{B} \implies b2 \in \# (\mathcal{B} - \{\#b1\}) \implies b1 \cap b2 = m$ 

sublocale symmetric-bibd  $\subseteq$  const-intersect-design  $\mathcal{V} \mathcal{B}$   $\Lambda$ 
   $\langle proof \rangle$ 

context const-intersect-design
begin

  lemma inter-num-le-block-size:
    assumes bl  $\in \# \mathcal{B}$ 
    assumes b  $\geq 2$ 
    shows m  $\leq \text{card } bl$ 
   $\langle proof \rangle$ 

  lemma const-inter-multiplicity-one:
    assumes bl  $\in \# \mathcal{B}$ 
    assumes m  $< \text{card } bl$ 
    shows multiplicity bl = 1
   $\langle proof \rangle$ 

  lemma mult-blocks-const-inter:
    assumes bl  $\in \# \mathcal{B}$ 
    assumes multiplicity bl > 1
    assumes b  $\geq 2$ 
    shows m = card bl
   $\langle proof \rangle$ 

  lemma simple-const-inter-block-size: ( $\bigwedge bl. bl \in \# \mathcal{B} \implies m < \text{card } bl$ )  $\implies$  simple-design  $\mathcal{V} \mathcal{B}$ 
   $\langle proof \rangle$ 

  lemma simple-const-inter-iff:
    assumes b  $\geq 2$ 
    shows size {#bl  $\in \# \mathcal{B} . \text{card } bl = m$ }  $\leq 1 \longleftrightarrow$  simple-design  $\mathcal{V} \mathcal{B}$ 
   $\langle proof \rangle$ 

  lemma empty-inter-implies-rep-one:
    assumes m = 0
    assumes x  $\in \mathcal{V}$ 
    shows  $\mathcal{B} \text{ rep } x \leq 1$ 
   $\langle proof \rangle$ 

  lemma empty-inter-implies-b-lt-v:
    assumes m = 0

```

```

shows b ≤ v
⟨proof⟩

end

locale simple-const-intersect-design = const-intersect-design + simple-design
end

```

## 4 Incidence Vectors and Matrices

Incidence Matrices are an important representation for any incidence set system. The majority of basic definitions and properties proved in this theory are based on Stinson [8] and Colbourn [3].

```

theory Incidence-Matrices imports Design-Extras Matrix-Vector-Extras List-Index.List-Index
Design-Theory.Design-Isomorphisms
begin

```

### 4.1 Incidence Vectors

A function which takes an ordered list of points, and a block, returning a 0-1 vector  $v$  where there is a 1 in the  $i$ th position if point  $i$  is in that block

```

definition inc-vec-of :: 'a list ⇒ 'a set ⇒ ('b :: {ring-1}) vec where
inc-vec-of Vs bl ≡ vec (length Vs) (λ i . if (Vs ! i) ∈ bl then 1 else 0)

```

```

lemma inc-vec-one-zero-elems: setv (inc-vec-of Vs bl) ⊆ {0, 1}
⟨proof⟩

```

```

lemma finite-inc-vec-elems: finite (setv (inc-vec-of Vs bl))
⟨proof⟩

```

```

lemma inc-vec-elems-max-two: card (setv (inc-vec-of Vs bl)) ≤ 2
⟨proof⟩

```

```

lemma inc-vec-dim: dim-vec (inc-vec-of Vs bl) = length Vs
⟨proof⟩

```

```

lemma inc-vec-index: i < length Vs ⇒ inc-vec-of Vs bl $ i = (if (Vs ! i) ∈ bl
then 1 else 0)
⟨proof⟩

```

```

lemma inc-vec-index-one-iff: i < length Vs ⇒ inc-vec-of Vs bl $ i = 1 ↔ Vs
! i ∈ bl
⟨proof⟩

```

```

lemma inc-vec-index-zero-iff: i < length Vs ⇒ inc-vec-of Vs bl $ i = 0 ↔ Vs
! i ∉ bl

```

$\langle proof \rangle$

```
lemma inc-vec-of-bij-betw:
  assumes inj-on f (set Vs)
  assumes bl ⊆ (set Vs)
  shows inc-vec-of Vs bl = inc-vec-of (map f Vs) (f ` bl)
⟨proof⟩
```

## 4.2 Incidence Matrices

A function which takes a list of points, and list of sets of points, and returns a  $v \times b$  0-1 matrix  $M$ , where  $v$  is the number of points, and  $b$  the number of sets, such that there is a 1 in the  $i, j$  position if and only if point  $i$  is in block  $j$ . The matrix has type ' $b$  mat' to allow for operations commonly used on matrices [8]

```
definition inc-mat-of :: 'a list ⇒ 'a set list ⇒ ('b :: {ring-1}) mat where
  inc-mat-of Vs Bs ≡ mat (length Vs) (length Bs) (λ (i,j) . if (Vs ! i) ∈ (Bs ! j) then 1 else 0)
```

Basic lemmas on the *inc-mat-of* matrix result (elements/dimensions/indexing)

```
lemma inc-mat-one-zero-elems: elements-mat (inc-mat-of Vs Bs) ⊆ {0, 1}
⟨proof⟩
```

```
lemma fin-incidence-mat-elems: finite (elements-mat (inc-mat-of Vs Bs))
⟨proof⟩
```

```
lemma inc-matrix-elems-max-two: card (elements-mat (inc-mat-of Vs Bs)) ≤ 2
⟨proof⟩
```

```
lemma inc-mat-of-index [simp]: i < dim-row (inc-mat-of Vs Bs) ⇒ j < dim-col (inc-mat-of Vs Bs) ⇒
  inc-mat-of Vs Bs $$ (i, j) = (if (Vs ! i) ∈ (Bs ! j) then 1 else 0)
⟨proof⟩
```

```
lemma inc-mat-dim-row: dim-row (inc-mat-of Vs Bs) = length Vs
⟨proof⟩
```

```
lemma inc-mat-dim-vec-row: dim-vec (row (inc-mat-of Vs Bs) i) = length Bs
⟨proof⟩
```

```
lemma inc-mat-dim-col: dim-col (inc-mat-of Vs Bs) = length Bs
⟨proof⟩
```

```
lemma inc-mat-dim-vec-col: dim-vec (col (inc-mat-of Vs Bs) i) = length Vs
⟨proof⟩
```

```
lemma inc-matrix-point-in-block-one: i < length Vs ⇒ j < length Bs ⇒ Vs ! i ∈ Bs ! j
```

$\implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-point-not-in-block-zero*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies Vs ! i \notin Bs ! j \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 0$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-point-in-block*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1 \implies Vs ! i \in Bs ! j$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-point-not-in-block*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 0 \implies Vs ! i \notin Bs ! j$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-point-not-in-block-iff*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 0 \longleftrightarrow Vs ! i \notin Bs ! j$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-point-in-block-iff*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1 \longleftrightarrow Vs ! i \in Bs ! j$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-subset-implies-one*:  
**assumes**  $I \subseteq \{\dots < \text{length } Vs\}$   
**assumes**  $j < \text{length } Bs$   
**assumes**  $(!) \text{ } Vs ^ I \subseteq Bs ! j$   
**assumes**  $i \in I$   
**shows**  $(\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-one-implies-membership*:  $I \subseteq \{\dots < \text{length } Vs\} \implies j < \text{length } Bs \implies (\bigwedge i. i \in I \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1) \implies i \in I \implies Vs ! i \in Bs ! j$   
 $\langle \text{proof} \rangle$

**lemma** *inc-matrix-elems-one-zero*:  $i < \text{length } Vs \implies j < \text{length } Bs \implies (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 0 \vee (\text{inc-mat-of } Vs \text{ } Bs) \text{ } \$\$ \text{ } (i, j) = 1$   
 $\langle \text{proof} \rangle$

Reasoning on Rows/Columns of the incidence matrix

**lemma** *inc-mat-col-def*:  $j < \text{length } Bs \implies i < \text{length } Vs \implies (\text{col } (\text{inc-mat-of } Vs \text{ } Bs) \text{ } j) \$ i = (\text{if } (Vs ! i \in Bs ! j) \text{ then } 1 \text{ else } 0)$   
 $\langle \text{proof} \rangle$

**lemma** *inc-mat-col-list-map-elem*:  $j < \text{length } Bs \implies i < \text{length } Vs \implies \text{col } (\text{inc-mat-of } Vs \text{ } Bs) \text{ } j \$ i = \text{map-vec } (\lambda x . \text{if } (x \in (Bs ! j)) \text{ then } 1 \text{ else } 0)$

*(vec-of-list Vs) \$ i*  
 *$\langle proof \rangle$*

**lemma** *inc-mat-col-list-map*:  $j < \text{length } Bs \implies \text{col}(\text{inc-mat-of } Vs \text{ } Bs) \ j = \text{map-vec}(\lambda x . \text{if } (x \in (Bs ! j)) \text{ then } 1 \text{ else } 0)$   
*(vec-of-list Vs)*  
 *$\langle proof \rangle$*

**lemma** *inc-mat-row-def*:  $j < \text{length } Bs \implies i < \text{length } Vs \implies \text{row}(\text{inc-mat-of } Vs \text{ } Bs) \ i \$ j = (\text{if } (Vs ! i \in Bs ! j) \text{ then } 1 \text{ else } 0)$   
 *$\langle proof \rangle$*

**lemma** *inc-mat-row-list-map-elem*:  $j < \text{length } Bs \implies i < \text{length } Vs \implies \text{row}(\text{inc-mat-of } Vs \text{ } Bs) \ i \$ j = \text{map-vec}(\lambda bl . \text{if } ((Vs ! i) \in bl) \text{ then } 1 \text{ else } 0)$   
*(vec-of-list Bs) \$ j*  
 *$\langle proof \rangle$*

**lemma** *inc-mat-row-list-map*:  $i < \text{length } Vs \implies \text{row}(\text{inc-mat-of } Vs \text{ } Bs) \ i = \text{map-vec}(\lambda bl . \text{if } ((Vs ! i) \in bl) \text{ then } 1 \text{ else } 0)$   
*(vec-of-list Bs)*  
 *$\langle proof \rangle$*

Connecting *inc-vec-of* and *inc-mat-of*

**lemma** *inc-mat-col-inc-vec*:  $j < \text{length } Bs \implies \text{col}(\text{inc-mat-of } Vs \text{ } Bs) \ j = \text{inc-vec-of } Vs \ (Bs ! j)$   
 *$\langle proof \rangle$*

**lemma** *inc-mat-of-cols-inc-vecs*:  $\text{cols}(\text{inc-mat-of } Vs \text{ } Bs) = \text{map}(\lambda j . \text{inc-vec-of } Vs \ j) \ Bs$   
 *$\langle proof \rangle$*

**lemma** *inc-mat-of-bij-betw*:  
**assumes** *inj-on f (set Vs)*  
**assumes**  $\bigwedge bl . bl \in (\text{set } Bs) \implies bl \subseteq (\text{set } Vs)$   
**shows** *inc-mat-of Vs Bs = inc-mat-of (map f Vs) (map ((\cdot) f) Bs)*  
 *$\langle proof \rangle$*

Definitions for the incidence matrix representation of common incidence system properties

**definition** *non-empty-col* ::  $('a :: \{\text{zero-neq-one}\}) \text{ mat} \Rightarrow \text{nat} \Rightarrow \text{bool}$  **where**  
*non-empty-col M j*  $\equiv \exists k. k \neq 0 \wedge k \in \$ \text{ col } M \ j$

**definition** *proper-inc-mat* ::  $('a :: \{\text{zero-neq-one}\}) \text{ mat} \Rightarrow \text{bool}$  **where**  
*proper-inc-mat M*  $\equiv (\text{dim-row } M > 0 \wedge \text{dim-col } M > 0)$

Matrix version of the representation number property (*rep*)

**definition** *mat-rep-num* ::  $('a :: \{\text{zero-neq-one}\}) \text{ mat} \Rightarrow \text{nat} \Rightarrow \text{nat}$  **where**  
*mat-rep-num M i*  $\equiv \text{count-vec}(\text{row } M \ i) \ 1$

Matrix version of the points index property (*index*)

```
definition mat-point-index :: ('a :: {zero-neq-one}) mat ⇒ nat set ⇒ nat where
mat-point-index M I ≡ card {j . j < dim-col M ∧ (∀ i ∈ I. M $$ (i, j) = 1)}
```

```
definition mat-inter-num :: ('a :: {zero-neq-one}) mat ⇒ nat ⇒ nat where
mat-inter-num M j1 j2 ≡ card {i . i < dim-row M ∧ M $$ (i, j1) = 1 ∧ M $$ (i, j2) = 1}
```

Matrix version of the block size property

```
definition mat-block-size :: ('a :: {zero-neq-one}) mat ⇒ nat ⇒ nat where
mat-block-size M j ≡ count-vec (col M j) 1
```

**lemma** non-empty-col-obtains:

```
assumes non-empty-col M j
obtains i where i < dim-row M and (col M j) $ i ≠ 0
⟨proof⟩
```

**lemma** non-empty-col-alt-def:

```
assumes j < dim-col M
shows non-empty-col M j ←→ (∃ i. i < dim-row M ∧ M $$ (i, j) ≠ 0)
⟨proof⟩
```

**lemma** proper-inc-mat-map: proper-inc-mat M ⇒ proper-inc-mat (map-mat f M)  
⟨proof⟩

**lemma** mat-point-index-alt: mat-point-index M I = card {j ∈ {0..<dim-col M} .  
(∀ i ∈ I . M \$\$ (i, j) = 1)}  
⟨proof⟩

**lemma** mat-block-size-sum-alt:

```
fixes M :: 'a :: {ring-1} mat
shows elements-mat M ⊆ {0, 1} ⇒ j < dim-col M ⇒ of-nat (mat-block-size
M j) = sum-vec (col M j)
⟨proof⟩
```

**lemma** mat-rep-num-sum-alt:

```
fixes M :: 'a :: {ring-1} mat
shows elements-mat M ⊆ {0, 1} ⇒ i < dim-row M ⇒ of-nat (mat-rep-num
M i) = sum-vec (row M i)
⟨proof⟩
```

**lemma** mat-point-index-two-alt:

```
assumes i1 < dim-row M
assumes i2 < dim-row M
shows mat-point-index M {i1, i2} = card {j . j < dim-col M ∧ M $$ (i1, j) = 1 ∧ M $$ (i2, j) = 1}
⟨proof⟩
```

Transpose symmetries

**lemma** trans-mat-rep-block-size-sym: j < dim-col M ⇒ mat-block-size M j =  
mat-rep-num M<sup>T</sup> j

$i < \text{dim-row } M \implies \text{mat-rep-num } M i = \text{mat-block-size } M^T i$   
 $\langle \text{proof} \rangle$

**lemma** *trans-mat-point-index-inter-sym*:  
 $i1 < \text{dim-row } M \implies i2 < \text{dim-row } M \implies \text{mat-point-index } M \{i1, i2\} = \text{mat-inter-num } M^T i1 i2$   
 $j1 < \text{dim-col } M \implies j2 < \text{dim-col } M \implies \text{mat-inter-num } M j1 j2 = \text{mat-point-index } M^T \{j1, j2\}$   
 $\langle \text{proof} \rangle$

### 4.3 0-1 Matrices

Incidence matrices contain only two elements: 0 and 1. We define a locale which provides a context to work in for matrices satisfying this condition for any '*b*' type.

```
locale zero-one-matrix =
  fixes matrix :: 'b :: {zero-neq-one} mat (M)
  assumes elems01: elements-mat M ⊆ {0, 1}
begin
```

Row and Column Properties of the Matrix

**lemma** *row-elems-ss01*:  $i < \text{dim-row } M \implies \text{vec-set}(\text{row } M i) \subseteq \{0, 1\}$   
 $\langle \text{proof} \rangle$

**lemma** *col-elems-ss01*:  
 assumes  $j < \text{dim-col } M$   
 shows  $\text{vec-set}(\text{col } M j) \subseteq \{0, 1\}$   
 $\langle \text{proof} \rangle$

**lemma** *col-nth-0-or-1-iff*:  
 assumes  $j < \text{dim-col } M$   
 assumes  $i < \text{dim-row } M$   
 shows  $\text{col } M j \$ i = 0 \longleftrightarrow \text{col } M j \$ i \neq 1$   
 $\langle \text{proof} \rangle$

**lemma** *row-nth-0-or-1-iff*:  
 assumes  $j < \text{dim-col } M$   
 assumes  $i < \text{dim-row } M$   
 shows  $\text{row } M i \$ j = 0 \longleftrightarrow \text{row } M i \$ j \neq 1$   
 $\langle \text{proof} \rangle$

**lemma** *transpose-entries*:  $\text{elements-mat } (M^T) \subseteq \{0, 1\}$   
 $\langle \text{proof} \rangle$

**lemma** *M-not-zero-simp*:  $j < \text{dim-col } M \implies i < \text{dim-row } M \implies M \$\$ (i, j) \neq 0$   
 $\implies M \$\$ (i, j) = 1$   
 $\langle \text{proof} \rangle$

**lemma** *M-not-one-simp*:  $j < \text{dim-col } M \implies i < \text{dim-row } M \implies M \$\$ (i, j) \neq 1 \implies M \$\$ (i, j) = 0$   
 $\langle \text{proof} \rangle$

Definition for mapping a column to a block

**definition** *map-col-to-block* ::  $'a :: \{\text{zero-neq-one}\}$  *vec*  $\Rightarrow \text{nat set where}$   
 $\text{map-col-to-block } c \equiv \{ i \in \{.. < \text{dim-vec } c\} . c \$ i = 1 \}$

**lemma** *map-col-to-block-alt*:  $\text{map-col-to-block } c = \{ i . i < \text{dim-vec } c \wedge c \$ i = 1 \}$   
 $\langle \text{proof} \rangle$

**lemma** *map-col-to-block-elem*:  $i < \text{dim-vec } c \implies i \in \text{map-col-to-block } c \longleftrightarrow c \$ i = 1$   
 $\langle \text{proof} \rangle$

**lemma** *in-map-col-valid-index*:  $i \in \text{map-col-to-block } c \implies i < \text{dim-vec } c$   
 $\langle \text{proof} \rangle$

**lemma** *map-col-to-block-size*:  $j < \text{dim-col } M \implies \text{card}(\text{map-col-to-block}(\text{col } M j)) = \text{mat-block-size } M j$   
 $\langle \text{proof} \rangle$

**lemma** *in-map-col-valid-index-M*:  $j < \text{dim-col } M \implies i \in \text{map-col-to-block}(\text{col } M j) \implies i < \text{dim-row } M$   
 $\langle \text{proof} \rangle$

**lemma** *map-col-to-block-elem-not*:  $c \in \text{set}(\text{cols } M) \implies i < \text{dim-vec } c \implies i \notin \text{map-col-to-block } c \longleftrightarrow c \$ i = 0$   
 $\langle \text{proof} \rangle$

**lemma** *obtain-block-index-map-block-set*:  
**assumes**  $bl \in \# \{ \# \text{map-col-to-block } c . c \in \# \text{mset}(\text{cols } M) \# \}$   
**obtains**  $j$  **where**  $j < \text{dim-col } M$  **and**  $bl = \text{map-col-to-block}(\text{col } M j)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-ord-inc-sys-point[simp]*:  $x < \text{dim-row } M \implies [0 .. < (\text{dim-row } M)] ! x = x$   
 $\langle \text{proof} \rangle$

**lemma** *mat-ord-inc-sys-block[simp]*:  $j < \text{dim-col } M \implies (\text{map}(\text{map-col-to-block})(\text{cols } M)) ! j = \text{map-col-to-block}(\text{col } M j)$   
 $\langle \text{proof} \rangle$

**lemma** *ordered-to-mset-col-blocks*:  
 $\{ \# \text{map-col-to-block } c . c \in \# \text{mset}(\text{cols } M) \# \} = \text{mset}(\text{map}(\text{map-col-to-block})(\text{cols } M))$   
 $\langle \text{proof} \rangle$

Lemmas on incidence matrix properties

**lemma** *non-empty-col-01*:

```

assumes  $j < \text{dim-col } M$ 
shows  $\text{non-empty-col } M j \longleftrightarrow 1 \in \$ \text{col } M j$ 
⟨proof⟩

lemma mat-rep-num-alt:
assumes  $i < \text{dim-row } M$ 
shows  $\text{mat-rep-num } M i = \text{card } \{j . j < \text{dim-col } M \wedge M \$\$ (i, j) = 1\}$ 
⟨proof⟩

lemma mat-rep-num-alt-col:  $i < \text{dim-row } M \implies \text{mat-rep-num } M i = \text{size } \{\# c$ 
 $\in \# (\text{mset } (\text{cols } M)) . c \$ i = 1\#\}$ 
⟨proof⟩

A zero one matrix is an incidence system

lemma map-col-to-block-wf:  $\bigwedge c. c \in \text{set } (\text{cols } M) \implies \text{map-col-to-block } c \subseteq \{0..<\text{dim-row } M\}$ 
⟨proof⟩

lemma one-implies-block-nempty:  $j < \text{dim-col } M \implies 1 \in \$ (\text{col } M j) \implies \text{map-col-to-block } (\text{col } M j) \neq \{\}$ 
⟨proof⟩

interpretation incidence-sys:  $\text{incidence-system } \{0..<\text{dim-row } M\}$ 
 $\{\# \text{map-col-to-block } c . c \in \# \text{mset } (\text{cols } M)\#\}$ 
⟨proof⟩

interpretation fin-incidence-sys:  $\text{finite-incidence-system } \{0..<\text{dim-row } M\}$ 
 $\{\# \text{map-col-to-block } c . c \in \# \text{mset } (\text{cols } M)\#\}$ 
⟨proof⟩

lemma block-nempty-implies-all-zeros:  $j < \text{dim-col } M \implies \text{map-col-to-block } (\text{col } M j) = \{\} \implies$ 
 $i < \text{dim-row } M \implies \text{col } M j \$ i = 0$ 
⟨proof⟩

lemma block-nempty-implies-no-one:  $j < \text{dim-col } M \implies \text{map-col-to-block } (\text{col } M j) = \{\} \implies \neg (1 \in \$ (\text{col } M j))$ 
⟨proof⟩

lemma mat-is-design:
assumes  $\bigwedge j. j < \text{dim-col } M \implies 1 \in \$ (\text{col } M j)$ 
shows  $\text{design } \{0..<\text{dim-row } M\} \{\# \text{map-col-to-block } c . c \in \# \text{mset } (\text{cols } M)\#\}$ 
⟨proof⟩

lemma mat-is-proper-design:
assumes  $\bigwedge j. j < \text{dim-col } M \implies 1 \in \$ (\text{col } M j)$ 
assumes  $\text{dim-col } M > 0$ 
shows  $\text{proper-design } \{0..<\text{dim-row } M\} \{\# \text{map-col-to-block } c . c \in \# \text{mset } (\text{cols } M)\#\}$ 

```

$\langle proof \rangle$

Show the 01 injective function preserves system properties

**lemma** *inj-on-01-hom-index*:

**assumes** *inj-on-01-hom f*  
**assumes**  $i < \text{dim-row } M$   $j < \text{dim-col } M$   
**shows**  $M \$\$ (i, j) = 1 \longleftrightarrow (\text{map-mat } f M) \$\$ (i, j) = 1$   
**and**  $M \$\$ (i, j) = 0 \longleftrightarrow (\text{map-mat } f M) \$\$ (i, j) = 0$

$\langle proof \rangle$

**lemma** *preserve-non-empty*:

**assumes** *inj-on-01-hom f*  
**assumes**  $j < \text{dim-col } M$   
**shows** *non-empty-col M j*  $\longleftrightarrow$  *non-empty-col (map-mat f M) j*

$\langle proof \rangle$

**lemma** *preserve-mat-rep-num*:

**assumes** *inj-on-01-hom f*  
**assumes**  $i < \text{dim-row } M$   
**shows** *mat-rep-num M i* = *mat-rep-num (map-mat f M) i*

$\langle proof \rangle$

**lemma** *preserve-mat-block-size*:

**assumes** *inj-on-01-hom f*  
**assumes**  $j < \text{dim-col } M$   
**shows** *mat-block-size M j* = *mat-block-size (map-mat f M) j*

$\langle proof \rangle$

**lemma** *preserve-mat-point-index*:

**assumes** *inj-on-01-hom f*  
**assumes**  $\bigwedge i. i \in I \implies i < \text{dim-row } M$   
**shows** *mat-point-index M I* = *mat-point-index (map-mat f M) I*

$\langle proof \rangle$

**lemma** *preserve-mat-inter-num*:

**assumes** *inj-on-01-hom f*  
**assumes**  $j1 < \text{dim-col } M$   $j2 < \text{dim-col } M$   
**shows** *mat-inter-num M j1 j2* = *mat-inter-num (map-mat f M) j1 j2*

$\langle proof \rangle$

**lemma** *lift-mat-01-index-iff*:

$i < \text{dim-row } M \implies j < \text{dim-col } M \implies (\text{lift-01-mat } M) \$\$ (i, j) = 0 \longleftrightarrow M \$\$ (i, j) = 0$   
 $i < \text{dim-row } M \implies j < \text{dim-col } M \implies (\text{lift-01-mat } M) \$\$ (i, j) = 1 \longleftrightarrow M \$\$ (i, j) = 1$

$\langle proof \rangle$

**lemma** *lift-mat-elems*: *elements-mat (lift-01-mat M)*  $\subseteq \{0, 1\}$

$\langle proof \rangle$

**lemma** *lift-mat-is-0-1*: zero-one-matrix (*lift-01-mat M*)  
 $\langle proof \rangle$

**lemma** *lift-01-mat-distinct-cols*: distinct (*cols M*)  $\implies$  distinct (*cols (lift-01-mat M)*)  
 $\langle proof \rangle$

**end**

Some properties must be further restricted to matrices having a '*a*' type  
**locale** *zero-one-matrix-ring-1* = zero-one-matrix *M* **for** *M* :: '*b*' :: {ring-1} mat  
**begin**

**lemma** *map-col-block-eq*:  
  **assumes** *c*  $\in$  set(*cols M*)  
  **shows** inc-vec-of [0.. $<$ dim-vec *c*] (*map-col-to-block c*) = *c*  
 $\langle proof \rangle$

**lemma** *inc-mat-of-map-rev*: inc-mat-of [0.. $<$ dim-row *M*] (*map map-col-to-block (cols M)*) = *M*  
 $\langle proof \rangle$

**lemma** *M-index-square-itself*:  $j < \text{dim-col } M \implies i < \text{dim-row } M \implies (M \$\$ (i, j)) \wedge 2 = M \$\$ (i, j)$   
 $\langle proof \rangle$

**lemma** *M-col-index-square-itself*:  $j < \text{dim-col } M \implies i < \text{dim-row } M \implies ((\text{col } M j) \$ i) \wedge 2 = (\text{col } M j) \$ i$   
 $\langle proof \rangle$

Scalar Prod Alternative definitions for matrix properties

**lemma** *scalar-prod-inc-vec-block-size-mat*:  
  **assumes**  $j < \text{dim-col } M$   
  **shows** (*col M j*)  $\cdot$  (*col M j*) = of-nat (*mat-block-size M j*)  
 $\langle proof \rangle$

**lemma** *scalar-prod-inc-vec-mat-inter-num*:  
  **assumes**  $j1 < \text{dim-col } M$   $j2 < \text{dim-col } M$   
  **shows** (*col M j1*)  $\cdot$  (*col M j2*) = of-nat (*mat-inter-num M j1 j2*)  
 $\langle proof \rangle$

**end**

Any matrix generated by *inc-mat-of* is a 0-1 matrix.

**lemma** *inc-mat-of-01-mat*: zero-one-matrix-ring-1 (*inc-mat-of Vs Bs*)  
 $\langle proof \rangle$

## 4.4 Ordered Incidence Systems

We impose an arbitrary ordering on the point set and block collection to enable matrix reasoning. Note that this is also common in computer algebra representations of designs

```
locale ordered-incidence-system =
  fixes  $\mathcal{V}s :: 'a list$  and  $\mathcal{B}s :: 'a set list$ 
  assumes wf-list:  $b \in \#(\text{mset } \mathcal{B}s) \implies b \subseteq \text{set } \mathcal{V}s$ 
  assumes distinct:  $\text{distinct } \mathcal{V}s$ 
```

An ordered incidence system, as it is defined on lists, can only represent finite incidence systems

```
sublocale ordered-incidence-system  $\subseteq$  finite-incidence-system set  $\mathcal{V}s$  mset  $\mathcal{B}s$ 
  ⟨proof⟩
```

```
lemma ordered-incidence-sysI:
  assumes finite-incidence-system  $\mathcal{V} \mathcal{B}$ 
  assumes  $\mathcal{V}s \in \text{permutations-of-set } \mathcal{V}$  and  $\mathcal{B}s \in \text{permutations-of-multiset } \mathcal{B}$ 
  shows ordered-incidence-system  $\mathcal{V}s \mathcal{B}s$ 
  ⟨proof⟩
```

```
lemma ordered-incidence-sysII:
  assumes finite-incidence-system  $\mathcal{V} \mathcal{B}$  and set  $\mathcal{V}s = \mathcal{V}$  and distinct  $\mathcal{V}s$  and mset
   $\mathcal{B}s = \mathcal{B}$ 
  shows ordered-incidence-system  $\mathcal{V}s \mathcal{B}s$ 
  ⟨proof⟩
```

```
context ordered-incidence-system
begin
```

For ease of notation, establish the same notation as for incidence systems

```
abbreviation  $\mathcal{V} \equiv \text{set } \mathcal{V}s$ 
abbreviation  $\mathcal{B} \equiv \text{mset } \mathcal{B}s$ 
```

Basic properties on ordered lists

```
lemma points-indexing:  $\mathcal{V}s \in \text{permutations-of-set } \mathcal{V}$ 
  ⟨proof⟩
```

```
lemma blocks-indexing:  $\mathcal{B}s \in \text{permutations-of-multiset } \mathcal{B}$ 
  ⟨proof⟩
```

```
lemma points-list-empty-iff:  $\mathcal{V}s = [] \longleftrightarrow \mathcal{V} = \{\}$ 
  ⟨proof⟩
```

```
lemma points-indexing-inj:  $\forall i \in I . i < \text{length } \mathcal{V}s \implies \text{inj-on } ((!) \mathcal{V}s) I$ 
  ⟨proof⟩
```

```
lemma blocks-list-empty-iff:  $\mathcal{B}s = [] \longleftrightarrow \mathcal{B} = \{\#\}$ 
  ⟨proof⟩
```

```

lemma blocks-list-nempty: proper-design  $\mathcal{V}$   $\mathcal{B} \implies \mathcal{B}s \neq []$ 
   $\langle proof \rangle$ 

lemma points-list-nempty: proper-design  $\mathcal{V}$   $\mathcal{B} \implies \mathcal{V}s \neq []$ 
   $\langle proof \rangle$ 

lemma points-list-length: length  $\mathcal{V}s = v$ 
   $\langle proof \rangle$ 

lemma blocks-list-length: length  $\mathcal{B}s = b$ 
   $\langle proof \rangle$ 

lemma valid-points-index:  $i < v \implies \mathcal{V}s ! i \in \mathcal{V}$ 
   $\langle proof \rangle$ 

lemma valid-points-index-cons:  $x \in \mathcal{V} \implies \exists i. \mathcal{V}s ! i = x \wedge i < v$ 
   $\langle proof \rangle$ 

lemma valid-points-index-obtains:
  assumes  $x \in \mathcal{V}$ 
  obtains  $i$  where  $\mathcal{V}s ! i = x \wedge i < v$ 
   $\langle proof \rangle$ 

lemma valid-blocks-index:  $j < b \implies \mathcal{B}s ! j \in \# \mathcal{B}$ 
   $\langle proof \rangle$ 

lemma valid-blocks-index-cons:  $bl \in \# \mathcal{B} \implies \exists j. \mathcal{B}s ! j = bl \wedge j < b$ 
   $\langle proof \rangle$ 

lemma valid-blocks-index-obtains:
  assumes  $bl \in \# \mathcal{B}$ 
  obtains  $j$  where  $\mathcal{B}s ! j = bl \wedge j < b$ 
   $\langle proof \rangle$ 

lemma block-points-valid-point-index:
  assumes  $bl \in \# \mathcal{B}$   $x \in bl$ 
  obtains  $i$  where  $i < \text{length } \mathcal{V}s \wedge \mathcal{V}s ! i = x$ 
   $\langle proof \rangle$ 

lemma points-set-index-img:  $\mathcal{V} = \text{image}(\lambda i . (\mathcal{V}s ! i)) (\{.. < v\})$ 
   $\langle proof \rangle$ 

lemma blocks-mset-image:  $\mathcal{B} = \text{image-mset} (\lambda i . (\mathcal{B}s ! i)) (\text{mset-set } \{.. < b\})$ 
   $\langle proof \rangle$ 

lemma incidence-cond-indexed[simp]:  $i < v \implies j < b \implies \text{incident } (\mathcal{V}s ! i) (\mathcal{B}s ! j) \longleftrightarrow (\mathcal{V}s ! i) \in (\mathcal{B}s ! j)$ 
   $\langle proof \rangle$ 

```

**lemma** *bij-betw-points-index*: *bij-betw* ( $\lambda i. \mathcal{V}s ! i$ )  $\{0..<\mathbf{v}\}$   $\mathcal{V}$   
 $\langle proof \rangle$

Some lemmas on cardinality due to different set descriptor filters

**lemma** *card-filter-point-indices*: *card*  $\{i \in \{0..<\mathbf{v}\}. P (\mathcal{V}s ! i)\} = \text{card } \{v \in \mathcal{V} . P v\}$   
 $\langle proof \rangle$

**lemma** *card-block-points-filter*:

**assumes**  $j < b$   
**shows** *card* ( $\mathcal{B}s ! j$ )  $= \text{card } \{i \in \{0..<\mathbf{v}\} . (\mathcal{V}s ! i) \in (\mathcal{B}s ! j)\}$   
 $\langle proof \rangle$

**lemma** *obtains-two-diff-block-indexes*:

**assumes**  $j1 < b$   
**assumes**  $j2 < b$   
**assumes**  $j1 \neq j2$   
**assumes**  $b \geq 2$   
**obtains**  $bl1 bl2$  **where**  $bl1 \in \# \mathcal{B}$  **and**  $\mathcal{B}s ! j1 = bl1$  **and**  $bl2 \in \# \mathcal{B} - \{\# bl1\}$   
**and**  $\mathcal{B}s ! j2 = bl2$   
 $\langle proof \rangle$

**lemma** *filter-size-blocks-eq-card-indexes*: *size*  $\{\# b \in \# \mathcal{B} . P b \#\} = \text{card } \{j \in \{..<(b)\}. P (\mathcal{B}s ! j)\}$   
 $\langle proof \rangle$

**lemma** *blocks-index-ne-belong*:

**assumes**  $i1 < \text{length } \mathcal{B}s$   
**assumes**  $i2 < \text{length } \mathcal{B}s$   
**assumes**  $i1 \neq i2$   
**shows**  $\mathcal{B}s ! i2 \in \# \mathcal{B} - \{\# (\mathcal{B}s ! i1)\}$   
 $\langle proof \rangle$

**lemma** *inter-num-points-filter-def*:

**assumes**  $j1 < b j2 < b j1 \neq j2$   
**shows** *card*  $\{x \in \{0..<\mathbf{v}\} . ((\mathcal{V}s ! x) \in (\mathcal{B}s ! j1) \wedge (\mathcal{V}s ! x) \in (\mathcal{B}s ! j2))\} = (\mathcal{B}s ! j1) \cap (\mathcal{B}s ! j2)$   
 $\langle proof \rangle$

Define an incidence matrix for this ordering of an incidence system

**abbreviation**  $N :: \text{int mat}$  **where**  
 $N \equiv \text{inc-mat-of } \mathcal{V}s \mathcal{B}s$

**sublocale** *zero-one-matrix-ring-1*  $N$   
 $\langle proof \rangle$

**lemma** *N-alt-def-dim*:  $N = \text{mat v b } (\lambda (i,j) . \text{if } (\text{incident } (\mathcal{V}s ! i) (\mathcal{B}s ! j)) \text{ then } 1 \text{ else } 0)$

$\langle proof \rangle$

Matrix Dimension related lemmas

**lemma**  $N\text{-carrier-mat}$ :  $N \in \text{carrier-mat}$  v b  
 $\langle proof \rangle$

**lemma**  $\text{dim-row-is-v}[\text{simp}]$ :  $\text{dim-row } N = \text{v}$   
 $\langle proof \rangle$

**lemma**  $\text{dim-col-is-b}[\text{simp}]$ :  $\text{dim-col } N = \text{b}$   
 $\langle proof \rangle$

**lemma**  $\text{dim-vec-row-N}$ :  $\text{dim-vec} (\text{row } N i) = \text{b}$   
 $\langle proof \rangle$

**lemma**  $\text{dim-vec-col-N}$ :  $\text{dim-vec} (\text{col } N i) = \text{v}$   $\langle proof \rangle$

**lemma**  $\text{dim-vec-N-col}$ :  
  **assumes**  $j < \text{b}$   
  **shows**  $\text{dim-vec} (\text{cols } N ! j) = \text{v}$   
 $\langle proof \rangle$

**lemma**  $N\text{-carrier-mat-01-lift}$ :  $\text{lift-01-mat } N \in \text{carrier-mat}$  v b  
 $\langle proof \rangle$

Transpose properties

**lemma**  $\text{transpose-N-mult-dim}$ :  $\text{dim-row} (N * N^T) = \text{v}$   $\text{dim-col} (N * N^T) = \text{v}$   
 $\langle proof \rangle$

**lemma**  $N\text{-trans-index-val}$ :  $i < \text{dim-col } N \implies j < \text{dim-row } N \implies$   
 $N^T \$\$ (i, j) = (\text{if } (\mathcal{V}s ! j) \in (\mathcal{B}s ! i) \text{ then } 1 \text{ else } 0)$   
 $\langle proof \rangle$

Matrix element and index related lemmas

**lemma**  $\text{mat-row-elems}$ :  $i < \text{v} \implies \text{vec-set} (\text{row } N i) \subseteq \{0, 1\}$   
 $\langle proof \rangle$

**lemma**  $\text{mat-col-elems}$ :  $j < \text{b} \implies \text{vec-set} (\text{col } N j) \subseteq \{0, 1\}$   
 $\langle proof \rangle$

**lemma**  $\text{matrix-elems-one-zero}$ :  $i < \text{v} \implies j < \text{b} \implies N \$\$ (i, j) = 0 \vee N \$\$ (i, j) = 1$   
 $\langle proof \rangle$

**lemma**  $\text{matrix-point-in-block-one}$ :  $i < \text{v} \implies j < \text{b} \implies (\mathcal{V}s ! i) \in (\mathcal{B}s ! j) \implies N \$\$ (i, j) = 1$   
 $\langle proof \rangle$

**lemma**  $\text{matrix-point-not-in-block-zero}$ :  $i < \text{v} \implies j < \text{b} \implies \mathcal{V}s ! i \notin \mathcal{B}s ! j \implies N \$\$ (i, j) = 0$

$\langle proof \rangle$

**lemma** *matrix-point-in-block*:  $i < v \implies j < b \implies N \$\$ (i, j) = 1 \implies \mathcal{V}s ! i \in \mathcal{B}s ! j$   
 $\langle proof \rangle$

**lemma** *matrix-point-not-in-block*:  $i < v \implies j < b \implies N \$\$ (i, j) = 0 \implies \mathcal{V}s ! i \notin \mathcal{B}s ! j$   
 $\langle proof \rangle$

**lemma** *matrix-point-not-in-block-iff*:  $i < v \implies j < b \implies N \$\$ (i, j) = 0 \longleftrightarrow \mathcal{V}s ! i \notin \mathcal{B}s ! j$   
 $\langle proof \rangle$

**lemma** *matrix-point-in-block-iff*:  $i < v \implies j < b \implies N \$\$ (i, j) = 1 \longleftrightarrow \mathcal{V}s ! i \in \mathcal{B}s ! j$   
 $\langle proof \rangle$

**lemma** *matrix-subset-implies-one*:  $I \subseteq \{.. < v\} \implies j < b \implies (!) \mathcal{V}s ' I \subseteq \mathcal{B}s ! j \implies i \in I \implies N \$\$ (i, j) = 1$   
 $\langle proof \rangle$

**lemma** *matrix-one-implements-membership*:  
 $I \subseteq \{.. < v\} \implies j < \text{size } \mathcal{B} \implies \forall i \in I. N \$\$ (i, j) = 1 \implies i \in I \implies \mathcal{V}s ! i \in \mathcal{B}s ! j$   
 $\langle proof \rangle$

Incidence Vector's of Incidence Matrix columns

**lemma** *col-inc-vec-of*:  $j < \text{length } \mathcal{B}s \implies \text{inc-vec-of } \mathcal{V}s (\mathcal{B}s ! j) = \text{col } N j$   
 $\langle proof \rangle$

**lemma** *inc-vec-eq-iff-blocks*:  
**assumes**  $bl \in \# \mathcal{B}$   
**assumes**  $bl' \in \# \mathcal{B}$   
**shows**  $\text{inc-vec-of } \mathcal{V}s bl = \text{inc-vec-of } \mathcal{V}s bl' \longleftrightarrow bl = bl'$   
 $\langle proof \rangle$

Incidence matrix column properties

**lemma** *N-col-def*:  $j < b \implies i < v \implies (\text{col } N j) \$ i = (\text{if } (\mathcal{V}s ! i \in \mathcal{B}s ! j) \text{ then } 1 \text{ else } 0)$   
 $\langle proof \rangle$

**lemma** *N-col-def-indiv*:  $j < b \implies i < v \implies \mathcal{V}s ! i \in \mathcal{B}s ! j \implies (\text{col } N j) \$ i = 1$   
 $j < b \implies i < v \implies \mathcal{V}s ! i \notin \mathcal{B}s ! j \implies (\text{col } N j) \$ i = 0$   
 $\langle proof \rangle$

**lemma** *N-col-list-map-elem*:  $j < b \implies i < v \implies \text{col } N j \$ i = \text{map-vec } (\lambda x . \text{if } (x \in (\mathcal{B}s ! j)) \text{ then } 1 \text{ else } 0) (\text{vec-of-list } \mathcal{V}s) \$ i$

$\langle proof \rangle$

**lemma**  $N\text{-col-list-map}$ :  $j < b \implies \text{col } N j = \text{map-vec } (\lambda x . \text{if } (x \in (\mathcal{B}s ! j)) \text{ then } 1 \text{ else } 0)$  ( $\text{vec-of-list } \mathcal{V}s$ )  
 $\langle proof \rangle$

**lemma**  $N\text{-col-mset-point-set-img}$ :  $j < b \implies \text{vec-mset } (\text{col } N j) = \text{image-mset } (\lambda x . \text{if } (x \in (\mathcal{B}s ! j)) \text{ then } 1 \text{ else } 0)$  ( $\text{mset-set } \mathcal{V}$ )  
 $\langle proof \rangle$

**lemma**  $\text{matrix-col-to-block}$ :  
**assumes**  $j < b$   
**shows**  $\mathcal{B}s ! j = (\lambda k . \mathcal{V}s ! k) ` \{i \in \{.. < v\} . (\text{col } N j) \$ i = 1\}$   
 $\langle proof \rangle$

**lemma**  $\text{matrix-col-to-block-v2}$ :  $j < b \implies \mathcal{B}s ! j = (\lambda k . \mathcal{V}s ! k) ` \text{map-col-to-block } (\text{col } N j)$   
 $\langle proof \rangle$

**lemma**  $\text{matrix-col-in-blocks}$ :  $j < b \implies (!) \mathcal{V}s ` \text{map-col-to-block } (\text{col } N j) \in \# \mathcal{B}$   
 $\langle proof \rangle$

**lemma**  $\text{inc-matrix-col-block}$ :  
**assumes**  $c \in \text{set } (\text{cols } N)$   
**shows**  $(\lambda x . \mathcal{V}s ! x) ` (\text{map-col-to-block } c) \in \# \mathcal{B}$   
 $\langle proof \rangle$

### Incidence Matrix Row Definitions

**lemma**  $N\text{-row-def}$ :  $j < b \implies i < v \implies (\text{row } N i) \$ j = (\text{if } (\mathcal{V}s ! i \in \mathcal{B}s ! j) \text{ then } 1 \text{ else } 0)$   
 $\langle proof \rangle$

**lemma**  $N\text{-row-list-map-elem}$ :  $j < b \implies i < v \implies \text{row } N i \$ j = \text{map-vec } (\lambda bl . \text{if } ((\mathcal{V}s ! i) \in bl) \text{ then } 1 \text{ else } 0)$  ( $\text{vec-of-list } \mathcal{B}s$ )  
 $\$ j$   
 $\langle proof \rangle$

**lemma**  $N\text{-row-list-map}$ :  $i < v \implies \text{row } N i = \text{map-vec } (\lambda bl . \text{if } ((\mathcal{V}s ! i) \in bl) \text{ then } 1 \text{ else } 0)$  ( $\text{vec-of-list } \mathcal{B}s$ )  
 $\langle proof \rangle$

**lemma**  $N\text{-row-mset-blocks-img}$ :  $i < v \implies \text{vec-mset } (\text{row } N i) = \text{image-mset } (\lambda x . \text{if } ((\mathcal{V}s ! i) \in x) \text{ then } 1 \text{ else } 0)$  ( $\mathcal{B}$ )  
 $\langle proof \rangle$

### Alternate Block representations

**lemma**  $\text{block-mat-cond-rep}$ :  
**assumes**  $j < \text{length } \mathcal{B}s$

**shows**  $(\mathcal{B}s ! j) = \{\mathcal{V}s ! i \mid i . i < \text{length } \mathcal{V}s \wedge N \$\$ (i, j) = 1\}$   
 $\langle \text{proof} \rangle$

**lemma** *block-mat-cond-rep'*:  $j < \text{length } \mathcal{B}s \implies (\mathcal{B}s ! j) = ((!) \mathcal{V}s) \cdot \{i . i < \text{length } \mathcal{V}s \wedge N \$\$ (i, j) = 1\}$   
 $\langle \text{proof} \rangle$

**lemma** *block-mat-cond-rev*:  
**assumes**  $j < \text{length } \mathcal{B}s$   
**shows**  $\{i . i < \text{length } \mathcal{V}s \wedge N \$\$ (i, j) = 1\} = ((\text{List-Index.index}) \mathcal{V}s) \cdot (\mathcal{B}s ! j)$   
 $\langle \text{proof} \rangle$

Incidence Matrix incidence system properties

**lemma** *incomplete-block-col*:  
**assumes**  $j < b$   
**assumes** *incomplete-block*  $(\mathcal{B}s ! j)$   
**shows**  $0 \in \$ (\text{col } N j)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-rep-num-N-row*:  
**assumes**  $i < v$   
**shows** *mat-rep-num*  $N i = \mathcal{B} \text{ rep } (\mathcal{V}s ! i)$   
 $\langle \text{proof} \rangle$

**lemma** *point-rep-mat-row-sum*:  $i < v \implies \text{sum-vec } (\text{row } N i) = \mathcal{B} \text{ rep } (\mathcal{V}s ! i)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-block-size-N-col*:  
**assumes**  $j < b$   
**shows** *mat-block-size*  $N j = \text{card } (\mathcal{B}s ! j)$   
 $\langle \text{proof} \rangle$

**lemma** *block-size-mat-rep-sum*:  $j < b \implies \text{sum-vec } (\text{col } N j) = \text{mat-block-size } N j$   
 $\langle \text{proof} \rangle$

**lemma** *mat-point-index-rep*:  
**assumes**  $I \subseteq \{.. < v\}$   
**shows** *mat-point-index*  $N I = \mathcal{B} \text{ index } ((\lambda i . \mathcal{V}s ! i) \cdot I)$   
 $\langle \text{proof} \rangle$

**lemma** *incidence-mat-two-index*:  $i1 < v \implies i2 < v \implies$   
 $\text{mat-point-index } N \{i1, i2\} = \mathcal{B} \text{ index } \{\mathcal{V}s ! i1, \mathcal{V}s ! i2\}$   
 $\langle \text{proof} \rangle$

**lemma** *ones-incidence-mat-block-size*:  
**assumes**  $j < b$   
**shows**  $((u_v v)_{v*} N) \$ j = \text{mat-block-size } N j$   
 $\langle \text{proof} \rangle$

**lemma** *mat-block-size-conv*:  $j < \text{dim-col } N \implies \text{card}(\mathcal{B}s ! j) = \text{mat-block-size } N j$   
 $\langle \text{proof} \rangle$

**lemma** *mat-inter-num-conv*:  
**assumes**  $j1 < \text{dim-col } N$   $j2 < \text{dim-col } N$   
**shows**  $(\mathcal{B}s ! j1) \cap (\mathcal{B}s ! j2) = \text{mat-inter-num } N j1 j2$   
 $\langle \text{proof} \rangle$

**lemma** *non-empty-col-map-conv*:  
**assumes**  $j < \text{dim-col } N$   
**shows**  $\text{non-empty-col } N j \longleftrightarrow \mathcal{B}s ! j \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma** *scalar-prod-inc-vec-inter-num*:  
**assumes**  $j1 < b$   $j2 < b$   
**shows**  $(\text{col } N j1) \cdot (\text{col } N j2) = (\mathcal{B}s ! j1) \cap (\mathcal{B}s ! j2)$   
 $\langle \text{proof} \rangle$

**lemma** *scalar-prod-block-size-lift-01*:  
**assumes**  $i < b$   
**shows**  $((\text{col } (\text{lift-01-mat } N) i) \cdot (\text{col } (\text{lift-01-mat } N) i)) = (\text{of-nat } (\text{card } (\mathcal{B}s ! i)))$   
 $:: ('b :: \{\text{ring-1}\})$   
 $\langle \text{proof} \rangle$

**lemma** *scalar-prod-inter-num-lift-01*:  
**assumes**  $j1 < b$   $j2 < b$   
**shows**  $((\text{col } (\text{lift-01-mat } N) j1) \cdot (\text{col } (\text{lift-01-mat } N) j2)) = (\text{of-nat } ((\mathcal{B}s ! j1) \cap (\mathcal{B}s ! j2))) :: ('b :: \{\text{ring-1}\})$   
 $\langle \text{proof} \rangle$

The System complement's incidence matrix flips 0's and 1's

**lemma** *map-block-complement-entry*:  $j < b \implies (\text{map block-complement } \mathcal{B}s) ! j = \text{block-complement } (\mathcal{B}s ! j)$   
 $\langle \text{proof} \rangle$

**lemma** *complement-mat-entries*:  
**assumes**  $i < v$  **and**  $j < b$   
**shows**  $(\mathcal{V}s ! i \notin \mathcal{B}s ! j) \longleftrightarrow (\mathcal{V}s ! i \in (\text{map block-complement } \mathcal{B}s) ! j)$   
 $\langle \text{proof} \rangle$

**lemma** *length-blocks-complement*:  $\text{length } (\text{map block-complement } \mathcal{B}s) = b$   
 $\langle \text{proof} \rangle$

**lemma** *ordered-complement*: *ordered-incidence-system*  $\mathcal{V}s$  ( $\text{map block-complement } \mathcal{B}s$ )  
 $\langle \text{proof} \rangle$

**interpretation** *ordered-comp*: *ordered-incidence-system*  $\mathcal{V}s$  ( $\text{map block-complement } \mathcal{B}s$ )

$\langle proof \rangle$

**lemma** complement-mat-entries-val:

assumes  $i < v$  and  $j < b$

shows ordered-comp.N \$\$ (i, j) = (\text{if } \mathcal{V}s ! i \in \mathcal{B}s ! j \text{ then } 0 \text{ else } 1)

$\langle proof \rangle$

**lemma** ordered-complement-mat: ordered-comp.N = mat v b ( $\lambda (i,j) . \text{if } (\mathcal{V}s ! i) \in (\mathcal{B}s ! j) \text{ then } 0 \text{ else } 1$ )

$\langle proof \rangle$

**lemma** ordered-complement-mat-map: ordered-comp.N = map-mat ( $\lambda x. \text{if } x = 1 \text{ then } 0 \text{ else } 1$ ) N

$\langle proof \rangle$

**end**

Establishing connection between incidence system and ordered incidence system locale

**lemma** (in incidence-system) alt-ordering-sysI:  $Vs \in \text{permutations-of-set } \mathcal{V} \implies Bs \in \text{permutations-of-multiset } \mathcal{B} \implies$

ordered-incidence-system Vs Bs

$\langle proof \rangle$

**lemma** (in finite-incidence-system) exists-ordering-sysI:  $\exists Vs Bs . Vs \in \text{permutations-of-set } \mathcal{V} \wedge$

$Bs \in \text{permutations-of-multiset } \mathcal{B} \wedge \text{ordered-incidence-system } Vs Bs$

$\langle proof \rangle$

**lemma** inc-sys-orderedI:

assumes incidence-system V B and distinct Vs and set Vs = V and mset Bs = B

shows ordered-incidence-system Vs Bs

$\langle proof \rangle$

Generalise the idea of an incidence matrix to an unordered context

**definition** is-incidence-matrix :: 'c :: {ring-1} mat  $\Rightarrow$  'a set  $\Rightarrow$  'a set multiset  $\Rightarrow$  bool where

is-incidence-matrix N V B  $\longleftrightarrow$

$(\exists Vs Bs . (Vs \in \text{permutations-of-set } V \wedge Bs \in \text{permutations-of-multiset } B \wedge N = (\text{inc-mat-of } Vs Bs)))$

**lemma** (in incidence-system) is-incidence-mat-alt: is-incidence-matrix N V B  $\longleftrightarrow$

$(\exists Vs Bs. (\text{set } Vs = \mathcal{V} \wedge \text{mset } Bs = \mathcal{B} \wedge \text{ordered-incidence-system } Vs Bs \wedge N = (\text{inc-mat-of } Vs Bs)))$

$\langle proof \rangle$

```

lemma (in ordered-incidence-system) is-incidence-mat-true: is-incidence-matrix N
  V B = True
  ⟨proof⟩

```

## 4.5 Incidence Matrices on Design Subtypes

```

locale ordered-design = ordered-incidence-system Vs Bs + design set Vs mset Bs
  for Vs and Bs
begin

lemma incidence-mat-non-empty-blocks:
  assumes j < b
  shows 1 ∈ (col N j)
  ⟨proof⟩

lemma all-cols-non-empty: j < dim-col N ==> non-empty-col N j
  ⟨proof⟩
end

locale ordered-simple-design = ordered-design Vs Bs + simple-design (set Vs) mset
  Bs for Vs Bs
begin

lemma block-list-distinct: distinct Bs
  ⟨proof⟩

lemma distinct-cols-N: distinct (cols N)
  ⟨proof⟩

lemma simp-blocks-length-card: length Bs = card (set Bs)
  ⟨proof⟩

lemma blocks-index-inj-on: inj-on (λ i . Bs ! i) {0..<length Bs}
  ⟨proof⟩

lemma x-in-block-set-img: assumes x ∈ set Bs shows x ∈ (!) Bs ‘ {0..<length Bs}
  ⟨proof⟩

lemma blocks-index-simp-bij-betw: bij-betw (λ i . Bs ! i) {0..<length Bs} (set Bs)
  ⟨proof⟩

lemma blocks-index-simp-unique: i1 < length Bs ==> i2 < length Bs ==> i1 ≠
  i2 ==> Bs ! i1 ≠ Bs ! i2
  ⟨proof⟩

lemma lift-01-distinct-cols-N: distinct (cols (lift-01-mat N))
  ⟨proof⟩

```

```

end

locale ordered-proper-design = ordered-design  $\mathcal{V}s \mathcal{B}s +$  proper-design set  $\mathcal{V}s mset$ 
 $\mathcal{B}s$ 
  for  $\mathcal{V}s$  and  $\mathcal{B}s$ 
begin

  lemma mat-is-proper: proper-inc-mat  $N$ 
     $\langle proof \rangle$ 

  end

  locale ordered-constant-rep = ordered-proper-design  $\mathcal{V}s \mathcal{B}s +$  constant-rep-design
  set  $\mathcal{V}s mset \mathcal{B}s r$ 
    for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $r$ 

  begin

    lemma incidence-mat-rep-num:  $i < v \implies$  mat-rep-num  $N i = r$ 
       $\langle proof \rangle$ 

    lemma incidence-mat-rep-num-sum:  $i < v \implies$  sum-vec (row  $N i) = r$ 
       $\langle proof \rangle$ 

    lemma transpose-N-mult-diag:
      assumes  $i = j$  and  $i < v$  and  $j < v$ 
      shows  $(N * N^T) \$\$ (i, j) = r$ 
       $\langle proof \rangle$ 

  end

  locale ordered-block-design = ordered-proper-design  $\mathcal{V}s \mathcal{B}s +$  block-design set  $\mathcal{V}s$ 
  mset  $\mathcal{B}s k$ 
    for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $k$ 

  begin

    lemma incidence-mat-block-size:  $j < b \implies$  mat-block-size  $N j = k$ 
       $\langle proof \rangle$ 

    lemma incidence-mat-block-size-sum:  $j < b \implies$  sum-vec (col  $N j) = k$ 
       $\langle proof \rangle$ 

    lemma ones-mult-incidence-mat-k-index:  $j < b \implies ((u_v v)_{v*} N) \$ j = k$ 
       $\langle proof \rangle$ 

    lemma ones-mult-incidence-mat-k:  $((u_v v)_{v*} N) = k \cdot_v (u_v b)$ 
       $\langle proof \rangle$ 

  
```

```

end

locale ordered-incomplete-design = ordered-block-design  $\mathcal{V}s \mathcal{B}s k + incomplete-design$ 
 $\mathcal{V} \mathcal{B} k$ 
for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $k$ 

begin

lemma incidence-mat-incomplete:  $j < b \implies 0 \in \$ (col N j)$ 
 $\langle proof \rangle$ 

end

locale ordered-t-wise-balance = ordered-proper-design  $\mathcal{V}s \mathcal{B}s + t\text{-wise-balance set}$ 
 $\mathcal{V}s mset \mathcal{B}s t \Lambda_t$ 
for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $t$  and  $\Lambda_t$ 

begin

lemma incidence-mat-des-index:
assumes  $I \subseteq \{0..v\}$ 
assumes card  $I = t$ 
shows mat-point-index  $N I = \Lambda_t$ 
 $\langle proof \rangle$ 

end

locale ordered-pairwise-balance = ordered-t-wise-balance  $\mathcal{V}s \mathcal{B}s 2 \Lambda + pairwise-balance$ 
set  $\mathcal{V}s mset \mathcal{B}s \Lambda$ 
for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $\Lambda$ 
begin

lemma incidence-mat-des-two-index:
assumes  $i_1 < v$ 
assumes  $i_2 < v$ 
assumes  $i_1 \neq i_2$ 
shows mat-point-index  $N \{i_1, i_2\} = \Lambda$ 
 $\langle proof \rangle$ 

lemma transpose-N-mult-off-diag:
assumes  $i \neq j$  and  $i < v$  and  $j < v$ 
shows  $(N * N^T) \$\$ (i, j) = \Lambda$ 
 $\langle proof \rangle$ 

end

context pairwise-balance
begin

```

```

lemma ordered-pbdI:
  assumes  $\mathcal{B} = mset \mathcal{B}s$  and  $\mathcal{V} = set \mathcal{V}s$  and distinct  $\mathcal{V}s$ 
  shows ordered-pairwise-balance  $\mathcal{V}s \mathcal{B}s \Lambda$ 
   $\langle proof \rangle$ 
end

locale ordered-regular-pairwise-balance = ordered-pairwise-balance  $\mathcal{V}s \mathcal{B}s \Lambda +$ 
  regular-pairwise-balance set  $\mathcal{V}s mset \mathcal{B}s \Lambda r$  for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $\Lambda$  and  $r$ 

sublocale ordered-regular-pairwise-balance  $\subseteq$  ordered-constant-rep
   $\langle proof \rangle$ 

context ordered-regular-pairwise-balance
begin

  Stinson's Theorem 1.15. Stinson [8] gives an iff condition for incidence
  matrices of regular pairwise balanced designs. The other direction is proven
  in the zero-one-matrix context

  lemma rpbd-incidence-matrix-cond:  $N * (N^T) = \Lambda \cdot_m (J_m v) + (r - \Lambda) \cdot_m (1_m v)$ 
   $\langle proof \rangle$ 
end

locale ordered-bibd = ordered-proper-design  $\mathcal{V}s \mathcal{B}s + bibd$  set  $\mathcal{V}s mset \mathcal{B}s k \Lambda$ 
  for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $k$  and  $\Lambda$ 

sublocale ordered-bibd  $\subseteq$  ordered-incomplete-design
   $\langle proof \rangle$ 

sublocale ordered-bibd  $\subseteq$  ordered-constant-rep  $\mathcal{V}s \mathcal{B}s r$ 
   $\langle proof \rangle$ 

sublocale ordered-bibd  $\subseteq$  ordered-pairwise-balance
   $\langle proof \rangle$ 

locale ordered-sym-bibd = ordered-bibd  $\mathcal{V}s \mathcal{B}s k \Lambda + symmetric-bibd$  set  $\mathcal{V}s mset \mathcal{B}s k \Lambda$ 
  for  $\mathcal{V}s$  and  $\mathcal{B}s$  and  $k$  and  $\Lambda$ 

sublocale ordered-sym-bibd  $\subseteq$  ordered-simple-design
   $\langle proof \rangle$ 

locale ordered-const-intersect-design = ordered-proper-design  $\mathcal{V}s \mathcal{B}s + const-intersect-design$ 
  set  $\mathcal{V}s mset \mathcal{B}s m$ 
  for  $\mathcal{V}s \mathcal{B}s m$ 

```

```

locale simp-ordered-const-intersect-design = ordered-const-intersect-design + ordered-simple-design
begin

lemma max-one-block-size-inter:
  assumes b ≥ 2
  assumes bl ∈# ℬ
  assumes card bl = m
  assumes bl2 ∈# ℬ - {#bl#}
  shows m < card bl2
⟨proof⟩

lemma block-size-inter-num-cases:
  assumes bl ∈# ℬ
  assumes b ≥ 2
  shows m < card bl ∨ (card bl = m ∧ (∀ bl' ∈# (ℬ - {#bl#}) . m < card bl'))
⟨proof⟩

lemma indexed-const-intersect:
  assumes j1 < b
  assumes j2 < b
  assumes j1 ≠ j2
  shows (ℬs ! j1) |∩| (ℬs ! j2) = m
⟨proof⟩

lemma const-intersect-block-size-diff:
  assumes j' < b and j < b and j ≠ j' and card (ℬs ! j') = m and b ≥ 2
  shows card (ℬs ! j) - m > 0
⟨proof⟩

lemma scalar-prod-inc-vec-const-inter:
  assumes j1 < b j2 < b j1 ≠ j2
  shows (col N j1) · (col N j2) = m
⟨proof⟩

end

```

## 4.6 Zero One Matrix Incidence System Existence

We prove 0-1 matrices with certain properties imply the existence of an incidence system with particular properties. This leads to Stinson's theorem in the other direction [8]

```

context zero-one-matrix
begin

```

```

lemma mat-is-ordered-incidence-sys: ordered-incidence-system [0..<(dim-row M)]
  (map (map-col-to-block) (cols M))
⟨proof⟩

```

```

interpretation mat-ord-inc-sys: ordered-incidence-system [0..<(dim-row M)] (map
(map-col-to-block) (cols M))
⟨proof⟩

lemma mat-ord-inc-sys-N: mat-ord-inc-sys.N = lift-01-mat M
⟨proof⟩

lemma map-col-to-block-mat-rep-num:
assumes x < dim-row M
shows ({# map-col-to-block c . c ∈ # mset (cols M)} rep x) = mat-rep-num
M x
⟨proof⟩

end

context zero-one-matrix-ring-1
begin

lemma transpose-cond-index-vals:
assumes M * (MT) = Λ ·m (Jm (dim-row M)) + (r - Λ) ·m (1m (dim-row
M))
assumes i < dim-row (M * (MT))
assumes j < dim-col (M * (MT))
shows i = j ⇒ (M * (MT)) $$ (i, j) = r i ≠ j ⇒ (M * (MT)) $$ (i, j) = Λ
⟨proof⟩

end

locale zero-one-matrix-int = zero-one-matrix-ring-1 M for M :: int mat
begin

    Some useful conditions on the transpose product for matrix system prop-
    erties

lemma transpose-cond-diag-r:
assumes i < dim-row (M * (MT))
assumes ∏ j. i = j ⇒ (M * (MT)) $$ (i, j) = r
shows mat-rep-num M i = r
⟨proof⟩

lemma transpose-cond-non-diag:
assumes i1 < dim-row (M * (MT))
assumes i2 < dim-row (M * (MT))
assumes i1 ≠ i2
assumes ∏ j. i. j ≠ i ⇒ i < dim-row (M * (MT)) ⇒ j < dim-row (M *
(MT)) ⇒ (M * (MT)) $$ (i, j) = Λ
shows Λ = mat-point-index M {i1, i2}
⟨proof⟩

```

**lemma** *trans-cond-implies-map-rep-num*:  
**assumes**  $M * (M^T) = \Lambda \cdot_m (J_m (\dim\text{-row } M)) + (r - \Lambda) \cdot_m (1_m (\dim\text{-row } M))$   
**assumes**  $x < \dim\text{-row } M$   
**shows** (*image-mset map-col-to-block (mset (cols M))*) *rep*  $x = r$   
*(proof)*

**lemma** *trans-cond-implies-map-index*:  
**assumes**  $M * (M^T) = \Lambda \cdot_m (J_m (\dim\text{-row } M)) + (r - \Lambda) \cdot_m (1_m (\dim\text{-row } M))$   
**assumes**  $ps \subseteq \{0..<\dim\text{-row } M\}$   
**assumes**  $\text{card } ps = 2$   
**shows** (*image-mset map-col-to-block (mset (cols M))*) *index*  $ps = \Lambda$   
*(proof)*

Stinson Theorem 1.15 existence direction

**lemma** *rpbexists*:  
**assumes**  $\dim\text{-row } M \geq 2$  — Min two points  
**assumes**  $\dim\text{-col } M \geq 1$  — Min one block  
**assumes**  $\bigwedge j. j < \dim\text{-col } M \implies 1 \in \text{col } M j$  — no empty blocks  
**assumes**  $M * (M^T) = \Lambda \cdot_m (J_m (\dim\text{-row } M)) + (r - \Lambda) \cdot_m (1_m (\dim\text{-row } M))$   
**shows** *ordered-regular-pairwise-balance*  $[0..<\dim\text{-row } M]$  (*map map-col-to-block (cols M)*)  $\Lambda r$   
*(proof)*

**lemma** *vec-k-uniform-mat-block-size*:  
**assumes**  $((u_v (\dim\text{-row } M)) \cdot_v M) = k \cdot_v (u_v (\dim\text{-col } M))$   
**assumes**  $j < \dim\text{-col } M$   
**shows** *mat-block-size*  $M j = k$   
*(proof)*

**lemma** *vec-k-impl-uniform-block-size*:  
**assumes**  $((u_v (\dim\text{-row } M)) \cdot_v M) = k \cdot_v (u_v (\dim\text{-col } M))$   
**assumes**  $bl \in \# (\text{image-mset map-col-to-block (mset (cols M))})$   
**shows**  $\text{card } bl = k$   
*(proof)*

**lemma** *bibd-exists*:  
**assumes**  $\dim\text{-col } M \geq 1$  — Min one block  
**assumes**  $\bigwedge j. j < \dim\text{-col } M \implies 1 \in \text{col } M j$  — no empty blocks  
**assumes**  $M * (M^T) = \Lambda \cdot_m (J_m (\dim\text{-row } M)) + (r - \Lambda) \cdot_m (1_m (\dim\text{-row } M))$   
**assumes**  $((u_v (\dim\text{-row } M)) \cdot_v M) = k \cdot_v (u_v (\dim\text{-col } M))$   
**assumes**  $(r :: \text{nat}) \geq 0$   
**assumes**  $k \geq 2$   $k < \dim\text{-row } M$   
**shows** *ordered-bibd*  $[0..<\dim\text{-row } M]$  (*map map-col-to-block (cols M)*)  $k \Lambda$   
*(proof)*

**end**

## 4.7 Isomorphisms and Incidence Matrices

If two incidence systems have the same incidence matrix, they are isomorphic. Similarly if two incidence systems are isomorphic there exists an ordering such that they have the same incidence matrix

**locale** *two-ordered-sys* = *D1*: ordered-incidence-system *Vs Bs* + *D2*: ordered-incidence-system *Vs' Bs'*

for *Vs* and *Bs* and *Vs'* and *Bs'*

**begin**

**lemma** *equal-inc-mat-isomorphism*:

assumes *D1.N = D2.N*

shows incidence-system-isomorphism *D1.V D1.B D2.V D2.B* ( $\lambda x . \mathcal{V}s'!$  (List-Index.index *Vs x*))

$\langle proof \rangle$

**lemma** *equal-inc-mat-isomorphism-ex*:  $D1.N = D2.N \implies \exists \pi . \text{incidence-system-isomorphism}$

*D1.V D1.B D2.V D2.B*  $\pi$

$\langle proof \rangle$

**lemma** *equal-inc-mat-isomorphism-obtain*:

assumes *D1.N = D2.N*

obtains  $\pi$  where incidence-system-isomorphism *D1.V D1.B D2.V D2.B*  $\pi$

$\langle proof \rangle$

**end**

**context** incidence-system-isomorphism

**begin**

**lemma** *exists-eq-inc-mats*:

assumes finite *V* finite *V'*

obtains *N* where is-incidence-matrix *N V B* and is-incidence-matrix *N V' B'*

$\langle proof \rangle$

**end**

**end**

## 5 Dual Systems

The concept of a dual incidence system [3] is an important property in design theory. It enables us to reason on the existence of several different types of design constructs through dual properties [8]

```
theory Dual-Systems imports Incidence-Matrices
begin
```

## 5.1 Dual Blocks

A dual design of  $(\mathcal{V}, \mathcal{B})$ , is the design where each block in  $\mathcal{B}$  represents a point  $x$ , and a block in a dual design is a set of blocks which  $x$  is in from the original design. It is important to note that if a block repeats in  $\mathcal{B}$ , each instance of the block is a distinct point. As such the definition below uses each block's list index as its identifier. The list of points would simply be the indices  $0..<\text{length } \mathcal{B}s$

```
definition dual-blocks :: 'a set  $\Rightarrow$  'a set list  $\Rightarrow$  nat set multiset where
dual-blocks  $\mathcal{V} \mathcal{B}s \equiv \{\# \{y . y < \text{length } \mathcal{B}s \wedge x \in \mathcal{B}s ! y\} . x \in \# (\text{mset-set } \mathcal{V})\# \}$ 
```

```
lemma dual-blocks-wf:  $b \in \# \text{dual-blocks } \mathcal{V} \mathcal{B}s \implies b \subseteq \{0..<\text{length } \mathcal{B}s\}$ 
⟨proof⟩
```

```
context ordered-incidence-system
begin
```

```
definition dual-blocks-ordered :: nat set list ( $\langle \mathcal{B}s^* \rangle$ ) where
dual-blocks-ordered  $\equiv \text{map } (\lambda x . \{y . y < \text{length } \mathcal{B}s \wedge x \in \mathcal{B}s ! y\}) \mathcal{V}s$ 
```

```
lemma dual-blocks-ordered-eq:  $\text{dual-blocks } \mathcal{V} \mathcal{B}s = \text{mset } (\mathcal{B}s^*)$ 
⟨proof⟩
```

```
lemma dual-blocks-len:  $\text{length } \mathcal{B}s^* = \text{length } \mathcal{V}s$ 
⟨proof⟩
```

A dual system is an incidence system

```
sublocale dual-sys: finite-incidence-system  $\{0..<\text{length } \mathcal{B}s\}$  dual-blocks  $\mathcal{V} \mathcal{B}s$ 
⟨proof⟩
```

```
lemma dual-is-ordered-inc-sys: ordered-incidence-system  $[0..<\text{length } \mathcal{B}s] \mathcal{B}s^*$ 
⟨proof⟩
```

```
interpretation ordered-dual-sys: ordered-incidence-system  $[0..<\text{length } \mathcal{B}s] \mathcal{B}s^*$ 
⟨proof⟩
```

## 5.2 Basic Dual Properties

```
lemma ord-dual-blocks-b: ordered-dual-sys.b = v
⟨proof⟩
```

```
lemma dual-blocks-b: dual-sys.b = v
⟨proof⟩
```

```
lemma dual-blocks-v: dual-sys.v = b
```

$\langle proof \rangle$

**lemma** *ord-dual-blocks-v*: *ordered-dual-sys.v = b*  
 $\langle proof \rangle$

**lemma** *dual-point-block*:  $i < v \implies \mathcal{B}s* ! i = \{y . y < \text{length } \mathcal{B}s \wedge (\mathcal{V}s ! i) \in \mathcal{B}s ! y\}$   
 $\langle proof \rangle$

**lemma** *dual-incidence-iff*:  $i < v \implies j < b \implies \mathcal{B}s ! j = bl \implies \mathcal{V}s ! i = x \implies (x \in bl \longleftrightarrow j \in \mathcal{B}s* ! i)$   
 $\langle proof \rangle$

**lemma** *dual-incidence-iff2*:  $i < v \implies j < b \implies (\mathcal{V}s ! i \in \mathcal{B}s ! j \longleftrightarrow j \in \mathcal{B}s* ! i)$   
 $\langle proof \rangle$

**lemma** *dual-blocks-point-exists*:  $bl \in \# \text{dual-blocks } \mathcal{V} \mathcal{B}s \implies \exists x . x \in \mathcal{V} \wedge bl = \{y . y < \text{length } \mathcal{B}s \wedge x \in \mathcal{B}s ! y\}$   
 $\langle proof \rangle$

**lemma** *dual-blocks-ne-index-ne*:  $j1 < \text{length } \mathcal{B}s* \implies j2 < \text{length } \mathcal{B}s* \implies \mathcal{B}s* ! j1 \neq \mathcal{B}s* ! j2 \implies j1 \neq j2$   
 $\langle proof \rangle$

**lemma** *dual-blocks-list-index-img*: *image-mset* ( $\lambda x . \mathcal{B}s* ! x$ ) (*mset-set* { $0..<\text{length } \mathcal{B}s*$ }) = *mset*  $\mathcal{B}s*$   
 $\langle proof \rangle$

**lemma** *dual-blocks-elem-iff*:  
**assumes**  $j < v$   
**shows**  $x \in (\mathcal{B}s* ! j) \longleftrightarrow \mathcal{V}s ! j \in \mathcal{B}s ! x \wedge x < b$   
 $\langle proof \rangle$

The incidence matrix of the dual of a design is just the transpose

**lemma** *dual-incidence-mat-eq-trans*: *ordered-dual-sys.N = N<sup>T</sup>*  
 $\langle proof \rangle$

**lemma** *dual-incidence-mat-eq-trans-rev*:  $(\text{ordered-dual-sys.N})^T = N$   
 $\langle proof \rangle$

### 5.3 Incidence System Dual Properties

Many common design properties have a dual in the dual design which enables extensive reasoning. Using incidence matrices and the transpose property these are easy to prove. We leave examples of counting proofs (commented out), to demonstrate how incidence matrices can significantly simplify reasoning.

**lemma** *dual-blocks-nempty*:

```

assumes ( $\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x > 0$ )
assumes  $bl \in \# \text{ dual-blocks } \mathcal{V} \mathcal{B}s$ 
shows  $bl \neq \{\}$ 
⟨proof⟩

lemma dual-blocks-size-is-rep:  $j < \text{length } \mathcal{B}s* \implies \text{card } (\mathcal{B}s* ! j) = \mathcal{B} \text{ rep } (\mathcal{V}s ! j)$ 
⟨proof⟩

lemma dual-blocks-size-is-rep-obtain:
assumes  $bl \in \# \text{ dual-blocks } \mathcal{V} \mathcal{B}s$ 
obtains  $x$  where  $x \in \mathcal{V}$  and  $\text{card } bl = \mathcal{B} \text{ rep } x$ 
⟨proof⟩

lemma dual-blocks-rep-is-size:
assumes  $i < \text{length } \mathcal{B}s$ 
shows  $(\text{mset } \mathcal{B}s*) \text{ rep } i = \text{card } (\mathcal{B}s ! i)$ 
⟨proof⟩

lemma dual-blocks-inter-index:
assumes  $j1 < \text{length } \mathcal{B}s* j2 < \text{length } \mathcal{B}s*$ 
shows  $(\mathcal{B}s* ! j1) \cap (\mathcal{B}s* ! j2) = \text{points-index } \mathcal{B} \{ \mathcal{V}s ! j1, \mathcal{V}s ! j2 \}$ 
⟨proof⟩

lemma dual-blocks-points-index-inter:
assumes  $i1 < b i2 < b$ 
shows  $(\text{mset } \mathcal{B}s*) \text{ index } \{i1, i2\} = (\mathcal{B}s ! i1) \cap (\mathcal{B}s ! i2)$ 
⟨proof⟩

end

```

## 5.4 Dual Properties for Design sub types

```

context ordered-design
begin

```

```

lemma dual-is-design:
assumes ( $\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x > 0$ ) — Required to ensure no blocks are
empty
shows design  $\{0..<\text{length } \mathcal{B}s\}$  (dual-blocks  $\mathcal{V} \mathcal{B}s$ )
⟨proof⟩
end

```

```

context ordered-proper-design

```

```

begin

lemma dual-sys-b-non-zero: dual-sys.b ≠ 0
⟨proof⟩

lemma dual-is-proper-design:
assumes ( $\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x > 0$ ) — Required to ensure no blocks are empty
shows proper-design {0.. $<\text{length } \mathcal{B}s\}$  (dual-blocks  $\mathcal{V} \mathcal{B}s$ )
⟨proof⟩

end

context ordered-block-design
begin

lemma dual-blocks-const-rep:  $i \in \{0..<\text{length } \mathcal{B}s\} \implies (\text{mset } \mathcal{B}s*) \text{ rep } i = k$ 
⟨proof⟩

lemma dual-blocks-constant-rep-design:
assumes ( $\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} \text{ rep } x > 0$ )
shows constant-rep-design {0.. $<\text{length } \mathcal{B}s\}$  (dual-blocks  $\mathcal{V} \mathcal{B}s$ ) k
⟨proof⟩

end

context ordered-constant-rep
begin

lemma dual-blocks-const-size:  $j < \text{length } \mathcal{B}s* \implies \text{card } (\mathcal{B}s* ! j) = r$ 
⟨proof⟩

lemma dual-is-block-design: block-design {0.. $<\text{length } \mathcal{B}s\}$  (dual-blocks  $\mathcal{V} \mathcal{B}s$ ) r
⟨proof⟩

end

context ordered-pairwise-balance
begin

lemma dual-blocks-const-intersect:
assumes  $j1 < \text{length } \mathcal{B}s* j2 < \text{length } \mathcal{B}s*$ 
assumes  $j1 \neq j2$ 
shows  $(\mathcal{B}s* ! j1) \cap (\mathcal{B}s* ! j2) = \Lambda$ 
⟨proof⟩

lemma dual-is-const-intersect-des:
assumes  $\Lambda > 0$ 

```

**shows** *const-intersect-design*  $\{0..<(length \mathcal{B}s)\}$  (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ )  $\Lambda$   
 $\langle proof \rangle$

**lemma** *dual-is-simp-const-inter-des*:

**assumes**  $\Lambda > 0$

**assumes**  $\bigwedge bl. bl \in \# \mathcal{B} \implies incomplete-block bl$

**shows** *simple-const-intersect-design*  $\{0..<(length \mathcal{B}s)\}$  (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ )  $\Lambda$   
 $\langle proof \rangle$

**end**

**context** *ordered-const-intersect-design*

**begin**

**lemma** *dual-is-balanced*:

**assumes**  $ps \subseteq \{0..<(length \mathcal{B}s)\}$

**assumes**  $card ps = 2$

**shows** (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ ) *index ps* = m  
 $\langle proof \rangle$

**lemma** *dual-is-pbd*:

**assumes**  $(\bigwedge x . x \in \mathcal{V} \implies \mathcal{B} rep x > 0)$

**assumes**  $b \geq 2$

**shows** *pairwise-balance*  $\{0..<(length \mathcal{B}s)\}$  (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ ) m  
 $\langle proof \rangle$

**end**

**context** *ordered-sym-bibd*

**begin**

**lemma** *dual-is-balanced*:

**assumes**  $ps \subseteq \{0..<(length \mathcal{B}s)\}$

**assumes**  $card ps = 2$

**shows** (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ ) *index ps* =  $\Lambda$   
 $\langle proof \rangle$

**lemma** *dual-bibd*: *bibd*  $\{0..<(length \mathcal{B}s)\}$  (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ ) r  $\Lambda$   
 $\langle proof \rangle$

The dual of a BIBD must be symmetric

**lemma** *dual-bibd-symmetric*: *symmetric-bibd*  $\{0..<(length \mathcal{B}s)\}$  (*dual-blocks*  $\mathcal{V} \mathcal{B}s$ )

r  $\Lambda$

$\langle proof \rangle$

**end**

## 5.5 Generalise Dual Concept

The above formalisation relies on one translation of a dual design. However, any design with an ordering of points and blocks such that the matrix is the transpose of the original is a dual. The definition below encapsulates this concept. Additionally, we prove an isomorphism exists between the generated dual from *dual-blocks* and any design satisfying the is dual definition

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

**definition** *is-dual*:: '*b* list  $\Rightarrow$  '*b* set list  $\Rightarrow$  bool **where**  
 $is\text{-dual } Vs' Bs' \equiv ordered\text{-incidence\text{-}system } Vs' Bs' \wedge (inc\text{-mat\text{-}of } Vs' Bs' = N^T)$

**lemma** *is-dualI*:

**assumes** *ordered-incidence-system* *Vs' Bs'*  
**assumes** (*inc-mat-of* *Vs' Bs' = N<sup>T</sup>*)  
**shows** *is-dual Vs' Bs'*  
*{proof}*

**lemma** *is-dualD1*:

**assumes** *is-dual Vs' Bs'*  
**shows** (*inc-mat-of* *Vs' Bs' = N<sup>T</sup>*)  
*{proof}*

**lemma** *is-dualD2*:

**assumes** *is-dual Vs' Bs'*  
**shows** *ordered-incidence-system* *Vs' Bs'*  
*{proof}*

**lemma** *generated-is-dual*: *is-dual [0..<(length Bs)] Bs\**  
*{proof}*

**lemma** *is-dual-isomorphism-generated*:

**assumes** *is-dual Vs' Bs'*  
**shows**  $\exists \pi. incidence\text{-system\text{-}isomorphism } (set Vs') (mset Bs') (\{0..<(length Bs)\}) (dual\text{-blocks } \mathcal{V} Bs) \pi$   
*{proof}*

**interpretation** *ordered-dual-sys*: *ordered-incidence-system [0..<length Bs] Bs\**  
*{proof}*

Original system is dual of the dual

**lemma** *is-dual-rev*: *ordered-dual-sys.is-dual Vs Bs*  
*{proof}*

**end**

**end**

## 6 Rank Argument - General

General lemmas to enable reasoning using the rank argument. This is described by Godsil [5] and Bukh [2], both of whom present it as a foundational technique

```
theory Rank-Argument-General imports Dual-Systems Jordan-Normal-Form.Determinant
Jordan-Normal-Form.DL-Rank Jordan-Normal-Form.Ring-Hom-Matrix BenOr-Kozen-Reif.More-Matrix
begin
```

### 6.1 Row/Column Operations

Extensions to the existing elementary operations are made to enable reasoning on multiple operations at once, similar to mathematical literature

**lemma** *index-mat-addrow-basic* [*simp*]:

$$\begin{aligned} i < \text{dim-row } A \implies j < \text{dim-col } A \implies \text{addrow } a k l A \$\$ (i,j) = & (\text{if } k = i \text{ then} \\ & (a * (A \$\$ (l,j)) + (A \$\$ (i,j))) \text{ else } A \$\$ (i,j)) \\ i < \text{dim-row } A \implies j < \text{dim-col } A \implies \text{addrow } a i l A \$\$ (i,j) = & (a * (A \$\$ (l,j)) \\ + (A \$\$ (i,j))) \\ i < \text{dim-row } A \implies j < \text{dim-col } A \implies k \neq i \implies \text{addrow } a k l A \$\$ (i,j) = & A \\ \$\$ (i,j) \\ \text{dim-row } (\text{addrow } a k l A) = \text{dim-row } A \text{ dim-col } (\text{addrow } a k l A) = \text{dim-col } A \\ \langle \text{proof} \rangle \end{aligned}$$

Function to add a column to multiple other columns

```
fun add-col-to-multiple :: 'a :: semiring-1  $\Rightarrow$  nat list  $\Rightarrow$  nat  $\Rightarrow$  'a mat  $\Rightarrow$  'a mat
where
add-col-to-multiple a [] l A = A |
add-col-to-multiple a (k # ks) l A = (addcol a k l (add-col-to-multiple a ks l A))
```

Function to add a row to multiple other rows

```
fun add-row-to-multiple :: 'a :: semiring-1  $\Rightarrow$  nat list  $\Rightarrow$  nat  $\Rightarrow$  'a mat  $\Rightarrow$  'a mat
where
add-row-to-multiple a [] l A = A |
add-row-to-multiple a (k # ks) l A = (addrow a k l (add-row-to-multiple a ks l A))
```

Function to add multiple rows to a single row

```
fun add-multiple-rows :: 'a :: semiring-1  $\Rightarrow$  nat  $\Rightarrow$  nat list  $\Rightarrow$  'a mat  $\Rightarrow$  'a mat
where
add-multiple-rows a k [] A = A |
add-multiple-rows a k (l # ls) A = (addrow a k l (add-multiple-rows a k ls A))
```

Function to add multiple columns to a single col

```
fun add-multiple-cols :: 'a :: semiring-1  $\Rightarrow$  nat  $\Rightarrow$  nat list  $\Rightarrow$  'a mat  $\Rightarrow$  'a mat
where
add-multiple-cols a k [] A = A |
add-multiple-cols a k (l # ls) A = (addcol a k l (add-multiple-cols a k ls A))
```

Basic lemmas on dimension and indexing of resulting matrix from above functions

**lemma** *add-multiple-rows-dim* [*simp*]:  
*dim-row* (*add-multiple-rows* *a* *k ls A*) = *dim-row A*  
*dim-col* (*add-multiple-rows* *a* *k ls A*) = *dim-col A*  
*(proof)*

**lemma** *add-multiple-rows-index-unchanged* [*simp*]:  
 $i < \text{dim-row } A \implies j < \text{dim-col } A \implies k \neq i \implies \text{add-multiple-rows } a \ k \ \text{ls } A \ \$\$ (i,j) = A \ \$\$ (i,j)$   
*(proof)*

**lemma** *add-multiple-rows-index-eq*:  
**assumes**  $i < \text{dim-row } A$  **and**  $j < \text{dim-col } A$  **and**  $i \notin \text{set ls}$  **and**  $\bigwedge l . l \in \text{set ls} \implies l < \text{dim-row } A$   
**shows**  $\text{add-multiple-rows } a \ i \ \text{ls } A \ \$\$ (i,j) = (\sum l \leftarrow \text{ls}. \ a * A \ \$\$ (l,j)) + A \$\$ (i,j)$   
*(proof)*

**lemma** *add-multiple-rows-index-eq-bounds*:  
**assumes**  $i < \text{dim-row } A$  **and**  $j < \text{dim-col } A$  **and**  $i < \text{low} \vee i \geq \text{up}$  **and**  $\text{up} \leq \text{dim-row } A$   
**shows**  $\text{add-multiple-rows } a \ i \ [\text{low}..\text{up}] \ A \ \$\$ (i,j) = (\sum l = \text{low}..\text{up}. \ a * A \ \$\$ (l,j)) + A \$\$ (i,j)$   
*(proof)*

**lemma** *add-multiple-cols-dim* [*simp*]:  
*dim-row* (*add-multiple-cols* *a* *k ls A*) = *dim-row A*  
*dim-col* (*add-multiple-cols* *a* *k ls A*) = *dim-col A*  
*(proof)*

**lemma** *add-multiple-cols-index-unchanged* [*simp*]:  
 $i < \text{dim-row } A \implies j < \text{dim-col } A \implies k \neq j \implies \text{add-multiple-cols } a \ k \ \text{ls } A \ \$\$ (i,j) = A \ \$\$ (i,j)$   
*(proof)*

**lemma** *add-multiple-cols-index-eq*:  
**assumes**  $i < \text{dim-row } A$  **and**  $j < \text{dim-col } A$  **and**  $j \notin \text{set ls}$  **and**  $\bigwedge l . l \in \text{set ls} \implies l < \text{dim-col } A$   
**shows**  $\text{add-multiple-cols } a \ j \ \text{ls } A \ \$\$ (i,j) = (\sum l \leftarrow \text{ls}. \ a * A \ \$\$ (i,l)) + A \$\$ (i,j)$   
*(proof)*

**lemma** *add-multiple-cols-index-eq-bounds*:  
**assumes**  $i < \text{dim-row } A$  **and**  $j < \text{dim-col } A$  **and**  $j < \text{low} \vee j \geq \text{up}$  **and**  $\text{up} \leq \text{dim-col } A$   
**shows**  $\text{add-multiple-cols } a \ j \ [\text{low}..\text{up}] \ A \ \$\$ (i,j) = (\sum l = \text{low}..\text{up}. \ a * A \ \$\$ (i,l)) + A \$\$ (i,j)$   
*(proof)*

**lemma** *add-row-to-multiple-dim* [*simp*]:  
*dim-row* (*add-row-to-multiple* *a* *ks l A*) = *dim-row A*  
*dim-col* (*add-row-to-multiple* *a* *ks l A*) = *dim-col A*

$\langle proof \rangle$

**lemma** *add-row-to-multiple-index-unchanged* [*simp*]:

$i < dim\text{-}row A \implies j < dim\text{-}col A \implies i \notin set\ ks \implies add\text{-}row\text{-}to\text{-}multiple\ a\ ks\ l$

$A\ \$\$ (i,j) = A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-row-to-multiple-index-unchanged-bound*:

$i < dim\text{-}row A \implies j < dim\text{-}col A \implies i < low \implies i \geq up \implies$

$add\text{-}row\text{-}to\text{-}multiple\ a\ [low..<up]\ l\ A\ \$\$ (i,j) = A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-row-to-multiple-index-change*:

**assumes**  $i < dim\text{-}row A$  **and**  $j < dim\text{-}col A$  **and**  $i \in set\ ks$  **and**  $distinct\ ks$  **and**  $l \notin set\ ks$

**and**  $l < dim\text{-}row A$

**shows**  $add\text{-}row\text{-}to\text{-}multiple\ a\ ks\ l\ A\ \$\$ (i,j) = (a * A\ \$\$ (l,j)) + A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-row-to-multiple-index-change-bounds*:

**assumes**  $i < dim\text{-}row A$  **and**  $j < dim\text{-}col A$  **and**  $i \geq low$  **and**  $i < up$  **and**  $l < low \vee l \geq up$

**and**  $l < dim\text{-}row A$

**shows**  $add\text{-}row\text{-}to\text{-}multiple\ a\ [low..<up]\ l\ A\ \$\$ (i,j) = (a * A\ \$\$ (l,j)) + A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-col-to-multiple-dim* [*simp*]:

$dim\text{-}row\ (add\text{-}col\text{-}to\text{-}multiple\ a\ ks\ l\ A) = dim\text{-}row\ A$

$dim\text{-}col\ (add\text{-}col\text{-}to\text{-}multiple\ a\ ks\ l\ A) = dim\text{-}col\ A$

$\langle proof \rangle$

**lemma** *add-col-to-multiple-index-unchanged* [*simp*]:

$i < dim\text{-}row A \implies j < dim\text{-}col A \implies j \notin set\ ks \implies add\text{-}col\text{-}to\text{-}multiple\ a\ ks\ l\ A$

$\$\$ (i,j) = A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-col-to-multiple-index-unchanged-bound*:

$i < dim\text{-}row A \implies j < dim\text{-}col A \implies j < low \implies j \geq up \implies$

$add\text{-}col\text{-}to\text{-}multiple\ a\ [low..<up]\ l\ A\ \$\$ (i,j) = A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-col-to-multiple-index-change*:

**assumes**  $i < dim\text{-}row A$  **and**  $j < dim\text{-}col A$  **and**  $j \in set\ ks$  **and**  $distinct\ ks$  **and**  $l \notin set\ ks$

**and**  $l < dim\text{-}col A$

**shows**  $add\text{-}col\text{-}to\text{-}multiple\ a\ ks\ l\ A\ \$\$ (i,j) = (a * A\ \$\$ (i,l)) + A\ \$\$ (i,j)$

$\langle proof \rangle$

**lemma** *add-col-to-multiple-index-change-bounds*:  
**assumes**  $i < \text{dim-row } A$  **and**  $j < \text{dim-col } A$  **and**  $j \geq \text{low}$  **and**  $j < \text{up}$  **and**  $l < \text{low} \vee l \geq \text{up}$   
**and**  $l < \text{dim-col } A$   
**shows**  $\text{add-col-to-multiple } a [\text{low..}<\text{up}] l A \$\$ (i, j) = (a * A\$\$ (i, l)) + A\$\$ (i, j)$   
*(proof)*

Operations specifically on 1st row/column

**lemma** *add-first-row-to-multiple-index*:  
**assumes**  $i < \text{dim-row } M$  **and**  $j < \text{dim-col } M$   
**shows**  $i = 0 \implies (\text{add-row-to-multiple } a [1..\text{dim-row } M] 0 M) \$\$ (i, j) = M \$\$ (i, j)$   
**and**  $i \neq 0 \implies (\text{add-row-to-multiple } a [1..\text{dim-row } M] 0 M) \$\$ (i, j) = (a * M\$\$ (0, j)) + M\$\$ (i, j)$   
*(proof)*

**lemma** *add-all-cols-to-first*:  
**assumes**  $i < \text{dim-row } (M)$   
**assumes**  $j < \text{dim-col } (M)$   
**shows**  $j \neq 0 \implies \text{add-multiple-cols } 1 0 [1..\text{dim-col } M] M \$\$ (i, j) = M \$\$ (i, j)$   
**and**  $j = 0 \implies \text{add-multiple-cols } 1 0 [1..\text{dim-col } M] M \$\$ (i, j) = (\sum l = 1..\text{dim-col } M. M \$\$ (i, l)) + M\$\$ (i, 0)$   
*(proof)*

Lemmas on the determinant of the matrix under extended row/column operations

**lemma** *add-row-to-multiple-carrier*:  
 $A \in \text{carrier-mat } n n \implies \text{add-row-to-multiple } a \text{ ks } l A \in \text{carrier-mat } n n$   
*(proof)*

**lemma** *add-col-to-multiple-carrier*:  
 $A \in \text{carrier-mat } n n \implies \text{add-col-to-multiple } a \text{ ks } l A \in \text{carrier-mat } n n$   
*(proof)*

**lemma** *add-multiple-rows-carrier*:  
 $A \in \text{carrier-mat } n n \implies \text{add-multiple-rows } a k ls A \in \text{carrier-mat } n n$   
*(proof)*

**lemma** *add-multiple-cols-carrier*:  
 $A \in \text{carrier-mat } n n \implies \text{add-multiple-cols } a k ls A \in \text{carrier-mat } n n$   
*(proof)*

**lemma** *add-row-to-multiple-det*:  
**assumes**  $l \notin \text{set ks}$  **and**  $l < n$  **and**  $A \in \text{carrier-mat } n n$   
**shows**  $\det(\text{add-row-to-multiple } a \text{ ks } l A) = \det A$   
*(proof)*

**lemma** *add-col-to-multiple-det*:

**assumes**  $l \notin \text{set } ks$  **and**  $l < n$  **and**  $A \in \text{carrier-mat } n n$   
**shows**  $\det(\text{add-col-to-multiple } a \ ks \ l \ A) = \det A$   
 $\langle proof \rangle$

**lemma** *add-multiple-cols-det*:  
**assumes**  $k \notin \text{set } ls$  **and**  $\bigwedge l. l \in \text{set } ls \implies l < n$  **and**  $A \in \text{carrier-mat } n n$   
**shows**  $\det(\text{add-multiple-cols } a \ k \ ls \ A) = \det A$   
 $\langle proof \rangle$

**lemma** *add-multiple-rows-det*:  
**assumes**  $k \notin \text{set } ls$  **and**  $\bigwedge l. l \in \text{set } ls \implies l < n$  **and**  $A \in \text{carrier-mat } n n$   
**shows**  $\det(\text{add-multiple-rows } a \ k \ ls \ A) = \det A$   
 $\langle proof \rangle$

## 6.2 Rank and Linear Independence

**abbreviation**  $\text{rank } v M \equiv \text{vec-space.rank } v M$

Basic lemma: the rank of the multiplication of two matrices will be less than the minimum of the individual ranks of those matrices. This directly follows from an existing lemmas in the linear algebra library which show independently that the resulting matrices rank is less than either the right or left matrix rank in the product

**lemma** *rank-mat-mult-lt-min-rank-factor*:  
**fixes**  $A :: 'a :: \{\text{conjugatable-ordered-field}\} \text{ mat}$   
**assumes**  $A \in \text{carrier-mat } n m$   
**assumes**  $B \in \text{carrier-mat } m nc$   
**shows**  $\text{rank } n (A * B) \leq \min(\text{rank } n A) (\text{rank } m B)$   
 $\langle proof \rangle$

Rank Argument 1: Given two a  $x \times y$  matrix  $M$  where  $MM^T$  has rank  $x$ ,  $x \leq y$

**lemma** *rank-argument*:  
**fixes**  $M :: ('c :: \{\text{conjugatable-ordered-field}\}) \text{ mat}$   
**assumes**  $M \in \text{carrier-mat } x y$   
**assumes**  $\text{vec-space.rank } x (M * M^T) = x$   
**shows**  $x \leq y$   
 $\langle proof \rangle$

Generalise the rank argument to use the determinant. If the determinant of the matrix is non-zero, than it's rank must be equal to  $x$ . This removes the need for someone to use facts on rank in their proofs.

**lemma** *rank-argument-det*:  
**fixes**  $M :: ('c :: \{\text{conjugatable-ordered-field}\}) \text{ mat}$   
**assumes**  $M \in \text{carrier-mat } x y$   
**assumes**  $\det(M * M^T) \neq 0$   
**shows**  $x \leq y$   
 $\langle proof \rangle$

```
end
```

## 7 Linear Bound Argument - General

Lemmas to enable general reasoning using the linear bound argument for combinatorial proofs. Jukna [7] presents a good overview of the mathematical background this theory is based on and applications

```
theory Linear-Bound-Argument imports Incidence-Matrices Jordan-Normal-Form.DL-Rank
Jordan-Normal-Form.Ring-Hom-Matrix
begin
```

### 7.1 Vec Space Extensions

Simple extensions to the existing vector space locale on linear independence

```
context vec-space
begin
lemma lin-indpt-set-card-lt-dim:
  fixes A :: 'a vec set
  assumes A ⊆ carrier-vec n
  assumes lin-indpt A
  shows card A ≤ dim
  ⟨proof⟩

lemma lin-indpt-dim-col-lt-dim:
  fixes A :: 'a mat
  assumes A ∈ carrier-mat n nc
  assumes distinct (cols A)
  assumes lin-indpt (set (cols A))
  shows nc ≤ dim
  ⟨proof⟩

lemma lin-comb-imp-lin-dep-fin:
  fixes A :: 'a vec set
  assumes finite A
  assumes A ⊆ carrier-vec n
  assumes lincomb c A = 0_v n
  assumes ∃ a. a ∈ A ∧ c a ≠ 0
  shows lin-dep A
  ⟨proof⟩
```

While a trivial definition, this enables us to directly reference the definition outside of a locale context, as *lin-indpt* is an inherited definition

```
definition lin-indpt-vs:: 'a vec set ⇒ bool where
lin-indpt-vs A ↔ lin-indpt A
```

```

lemma lin-comb-sum-lin-indpt:
  fixes A :: 'a vec list
  assumes set (A) ⊆ carrier-vec n
  assumes distinct A
  assumes  $\bigwedge f. \text{lincomb-list} (\lambda i. f (A ! i)) A = 0_v n \implies \forall v \in (\text{set } A). f v = 0$ 
  shows lin-indpt (set A)
  ⟨proof⟩

lemma lin-comb-mat-lin-indpt:
  fixes A :: 'a vec list
  assumes set (A) ⊆ carrier-vec n
  assumes distinct A
  assumes  $\bigwedge f. \text{mat-of-cols} n A *_v \text{vec} (\text{length } A) (\lambda i. f (A ! i)) = 0_v n \implies \forall v \in (\text{set } A). f v = 0$ 
  shows lin-indpt (set A)
  ⟨proof⟩

lemma lin-comb-mat-lin-indpt-vs:
  fixes A :: 'a vec list
  assumes set (A) ⊆ carrier-vec n
  assumes distinct A
  assumes  $\bigwedge f. \text{mat-of-cols} n A *_v \text{vec} (\text{length } A) (\lambda i. f (A ! i)) = 0_v n \implies \forall v \in (\text{set } A). f v = 0$ 
  shows lin-indpt-vs (set A)
  ⟨proof⟩

end

```

## 7.2 Linear Bound Argument Lemmas

Three general representations of the linear bound argument, requiring a direct fact of linear independence on the rows of the vector space over either a set A of vectors, list xs of vectors or a Matrix' columns

```

lemma lin-bound-arg-general-set:
  fixes A :: ('a :: {field}) vec set
  assumes A ⊆ carrier-vec nr
  assumes vec-space.lin-indpt-vs nr A
  shows card A ≤ nr
  ⟨proof⟩

lemma lin-bound-arg-general-list:
  fixes xs :: ('a :: {field}) vec list
  assumes distinct xs
  assumes (set xs) ⊆ carrier-vec nr
  assumes vec-space.lin-indpt-vs nr (set xs)
  shows length (xs) ≤ nr
  ⟨proof⟩

```

```

lemma lin-bound-arg-general:
  fixes A ::('a :: {field}) mat
  assumes distinct (cols A)
  assumes A ∈ carrier-mat nr nc
  assumes vec-space.lin-indpt-vs nr (set (cols A))
  shows nc ≤ nr
  ⟨proof⟩

```

The linear bound argument lemma on a matrix requiring the lower level assumption on a linear combination. This removes the need to directly refer to any aspect of the linear algebra libraries

```

lemma lin-bound-argument:
  fixes A :: ('a :: {field}) mat
  assumes distinct (cols A)
  assumes A ∈ carrier-mat nr nc
  assumes  $\bigwedge f. A *_v \text{vec } nc (\lambda i. f (\text{col } A i)) = 0_v$  nr  $\implies \forall v \in (\text{set } (\text{cols } A)). f v = 0$ 
  shows nc ≤ nr
  ⟨proof⟩

```

A further extension to present the linear combination directly as a sum. This manipulation from vector product to a summation was found to commonly be repeated in proofs applying this rule

```

lemma lin-bound-argument2:
  fixes A :: ('a :: {field}) mat
  assumes distinct (cols A)
  assumes A ∈ carrier-mat nr nc
  assumes  $\bigwedge f. \text{vec } nr (\lambda i. \sum j \in \{0..<nc\} . f (\text{col } A j) * (\text{col } A j) \$ i) = 0_v$  nr
   $\implies \forall v \in (\text{set } (\text{cols } A)). f v = 0$ 
  shows nc ≤ nr
  ⟨proof⟩

```

end

## 8 Fisher's Inequality

This theory presents the proof of Fisher's Inequality [4] on BIBD's (i.e. uniform Fisher's) and the generalised nonuniform Fisher's Inequality

```

theory Fishers-Inequality imports Rank-Argument-General Linear-Bound-Argument
begin

```

### 8.1 Uniform Fisher's Inequality

```

context ordered-bibd
begin

```

Row/Column transformation steps

Following design theory lecture notes from MATH3301 at the University of Queensland [6], a simple transformation to produce an upper triangular matrix using elementary row operations is to (1) Subtract the first row from each other row, and (2) add all columns to the first column

```
lemma transform-N-step1-vals:
defines mdef:  $M \equiv (N * N^T)$ 
assumes  $i < \text{dim-row } M$ 
assumes  $j < \text{dim-col } M$ 
shows  $i = 0 \implies j = 0 \implies (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } M] 0 M)$ 
 $\$\$ (i, j) = (\text{int r}) — \text{top left elem}$ 
and  $i \neq 0 \implies j = 0 \implies (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } M] 0 M) \$\$$ 
 $(i, j) = (\text{int } \Lambda) - (\text{int r}) — \text{first column ex. 1}$ 
and  $i = 0 \implies j \neq 0 \implies (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } M] 0 M) \$\$$ 
 $(i, j) = (\text{int } \Lambda) — \text{first row ex. 1}$ 
and  $i \neq 0 \implies j \neq 0 \implies i = j \implies (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } M]$ 
 $0 M) \$\$ (i, j) = (\text{int r}) - (\text{int } \Lambda) — \text{diagonal ex. 1}$ 
and  $i \neq 0 \implies j \neq 0 \implies i \neq j \implies (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } M]$ 
 $0 M) \$\$ (i, j) = 0 — \text{everything else}$ 
<proof>

lemma transform-N-step2-vals:
defines mdef:  $M \equiv (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } (N * N^T)] 0 (N * N^T))$ 
assumes  $i < \text{dim-row } (M)$ 
assumes  $j < \text{dim-col } (M)$ 
shows  $i = 0 \implies j = 0 \implies \text{add-multiple-cols } 1 0 [1..<\text{dim-col } M] M \$\$ (i, j)$ 
 $=$ 
 $(\text{int r}) + (\text{int } \Lambda) * (\text{v} - 1) — \text{top left element}$ 
and  $i = 0 \implies j \neq 0 \implies \text{add-multiple-cols } 1 0 [1..<\text{dim-col } M] M \$\$ (i, j) =$ 
 $(\text{int } \Lambda) — \text{top row}$ 
and  $i \neq 0 \implies i = j \implies \text{add-multiple-cols } 1 0 [1..<\text{dim-col } M] M \$\$ (i, j) =$ 
 $(\text{int r}) - (\text{int } \Lambda) — \text{Diagonal}$ 
and  $i \neq 0 \implies i \neq j \implies \text{add-multiple-cols } 1 0 [1..<\text{dim-col } M] M \$\$ (i, j) =$ 
 $0 — \text{Everything else}$ 
<proof>
```

Transformed matrix is upper triangular

```
lemma transform-upper-triangular:
defines mdef:  $M \equiv (\text{add-row-to-multiple } (-1) [1..<\text{dim-row } (N * N^T)] 0 (N * N^T))$ 
shows  $\text{upper-triangular } (\text{add-multiple-cols } 1 0 [1..<\text{dim-col } M] M)$ 
<proof>
```

Find the determinant of the  $NN^T$  matrix using transformed matrix values

```
lemma determinant-inc-mat-square:  $\det (N * N^T) = (\text{r} + \Lambda * (\text{v} - 1)) * (\text{r} - \Lambda)^\text{v} (\text{v} - 1)$ 
<proof>
```

Fisher's Inequality using the rank argument. Note that to use the rank argument we must first map  $N$  to a real matrix. It is useful to explicitly include the parameters which should be used in the application of the *rank-argument-det* lemma

```
theorem Fishers-Inequality-BIBD:  $v \leq b$ 
  ⟨proof⟩
```

```
end
```

## 8.2 Generalised Fisher's Inequality

```
context simp-ordered-const-intersect-design
begin
```

Lemma to reason on sum coefficients

```
lemma sum-split-coeffs-0:
  fixes  $c :: nat \Rightarrow real$ 
  assumes  $b \geq 2$ 
  assumes  $m > 0$ 
  assumes  $j' < b$ 
  assumes  $0 = (\sum j \in \{0..<b\} . (c j)^{\wedge 2} * ((card (\mathcal{B}s ! j)) - (int m))) +$ 
     $m * ((\sum j \in \{0..<b\} . c j)^{\wedge 2})$ 
  shows  $c j' = 0$ 
  ⟨proof⟩
```

The general non-uniform version of fisher's inequality is also known as the "Block town problem". In this case we are working in a constant intersect design, hence the inequality is the opposite way around compared to the BIBD version. The theorem below is the more traditional set theoretic approach. This proof is based off one by Jukna [7]

```
theorem general-fishers-inequality:  $b \leq v$ 
  ⟨proof⟩
```

```
end
```

Using the dual design concept, it is easy to translate the set theoretic general definition of Fisher's inequality to a more traditional design theoretic version on pairwise balanced designs. Two versions of this are given using different trivial (but crucial) conditions on design properties

```
context ordered-pairwise-balance
begin
```

```
corollary general-nonuniform-fishers: — only valid on incomplete designs
  assumes  $\Lambda > 0$ 
  assumes  $\bigwedge bl. bl \in \# \mathcal{B} \implies incomplete-block bl$ 
  — i.e. not a super trivial design with only complete blocks
  shows  $v \leq b$ 
  ⟨proof⟩
```

```

corollary general-nonuniform-fishers-comp:
  assumes  $\Lambda > 0$ 
  assumes count  $\mathcal{B} \mathcal{V} < \Lambda$  — i.e. not a super trivial design with only complete
  blocks and single blocks
  shows  $v \leq b$ 
   $\langle proof \rangle$ 

end
end

```

## 9 Matrices/Vectors mod x

This section formalises operations and properties mod some integer x on integer matrices and vectors. Much of this file was no longer needed for fishers once the generic idea of lifting a 0-1 matrix was introduced, however it is left as an example for future work on matrices mod n, beyond 0-1 matrices

```

theory Vector-Matrix-Mod imports Matrix-Vector-Extras
Berlekamp-Zassenhaus.Finite-Field Berlekamp-Zassenhaus.More-Missing-Multiset
begin

```

Simple abbreviations for main mapping functions

```

abbreviation to-int-mat :: ' $a :: finite\ mod-ring$  mat  $\Rightarrow int\ mat$ ' where
  to-int-mat  $\equiv$  map-mat to-int-mod-ring

```

```

abbreviation to-int-vec :: ' $a :: finite\ mod-ring$  vec  $\Rightarrow int\ vec$ ' where
  to-int-vec  $\equiv$  map-vec to-int-mod-ring

```

```

interpretation of-int-mod-ring-hom-sr: semiring-hom of-int-mod-ring
 $\langle proof \rangle$ 

```

NOTE: The context directly below is copied from Matrix Vector Extras, as for some reason they can't be used locally if not? Ideally remove in future if possible

```

context inj-zero-hom
begin

```

```

lemma vec-hom-zero-iff[simp]:  $(map\text{-}vec\ hom\ x = 0_v\ n) = (x = 0_v\ n)$ 
 $\langle proof \rangle$ 

```

```

lemma mat-hom-inj: map-mat hom A = map-mat hom B  $\implies A = B$ 
 $\langle proof \rangle$ 

```

```

lemma vec-hom-inj: map-vec hom v = map-vec hom w  $\implies v = w$ 
 $\langle proof \rangle$ 

```

```

lemma vec-hom-set-distinct-iff:
  fixes xs :: 'a vec list
  shows distinct xs  $\longleftrightarrow$  distinct (map (map-vec hom) xs)
   $\langle proof \rangle$ 
end

```

## 9.1 Basic Mod Context

```

locale mat-mod = fixes m :: int
assumes non-triv-m: m > 1
begin

```

First define the mod operations on vectors

```

definition vec-mod :: int vec  $\Rightarrow$  int vec where
vec-mod v  $\equiv$  map-vec ( $\lambda$  x . x mod m) v

```

```

lemma vec-mod-dim[simp]: dim-vec (vec-mod v) = dim-vec v
 $\langle proof \rangle$ 

```

```

lemma vec-mod-index[simp]: i < dim-vec v  $\Longrightarrow$  (vec-mod v) $ i = (v \$ i) mod m
 $\langle proof \rangle$ 

```

```

lemma vec-mod-index-same[simp]: i < dim-vec v  $\Longrightarrow$  v $ i < m  $\Longrightarrow$  v $ i  $\geq$  0
 $\Longrightarrow$  (vec-mod v) $ i = v $ i
 $\langle proof \rangle$ 

```

```

lemma vec-setI2: i < dim-vec v  $\Longrightarrow$  v $ i  $\in$  setv v
 $\langle proof \rangle$ 

```

```

lemma vec-mod-eq: setv v  $\subseteq$  {0..<m}  $\Longrightarrow$  vec-mod v = v
 $\langle proof \rangle$ 

```

```

lemma vec-mod-set-vec-same:setv v  $\subseteq$  {0..<m}  $\Longrightarrow$  setv (vec-mod v) = setv v
 $\langle proof \rangle$ 

```

Define the mod operation on matrices

```

definition mat-mod :: int mat  $\Rightarrow$  int mat where
mat-mod M  $\equiv$  map-mat ( $\lambda$  x. x mod m) M

```

```

lemma mat-mod-dim[simp]: dim-row (mat-mod M) = dim-row M dim-col (mat-mod M) = dim-col M
 $\langle proof \rangle$ 

```

```

lemma mat-mod-index [simp]: i < dim-row M  $\Longrightarrow$  j < dim-col M  $\Longrightarrow$  (mat-mod M) $$ (i, j) = (M $$ (i, j)) mod m
 $\langle proof \rangle$ 

```

**lemma** *mat-mod-index-same*[simp]:  $i < \text{dim-row } M \implies j < \text{dim-col } M \implies M \$\$ (i, j) < m \implies M \$\$ (i, j) \geq 0 \implies \text{mat-mod } M \$\$ (i, j) = M \$\$ (i, j)$   
 $\langle \text{proof} \rangle$

**lemma** *elements-matI2*:  $i < \text{dim-row } A \implies j < \text{dim-col } A \implies A \$\$ (i, j) \in \text{elements-mat } A$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-elements-in*:  
**assumes**  $x \in \text{elements-mat } M$   
**shows**  $x \text{ mod } m \in \text{elements-mat} (\text{mat-mod } M)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-elements-map*:  
**assumes**  $x \in \text{elements-mat } M$   
**shows**  $\text{elements-mat} (\text{mat-mod } M) = (\lambda x. x \text{ mod } m) ` (\text{elements-mat } M)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-eq-cond*:  
**assumes**  $\text{elements-mat } M \subseteq \{0..<m\}$   
**shows**  $\text{mat-mod } M = M$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-eq-elements-cond*:  $\text{elements-mat } M \subseteq \{0..<m\} \implies \text{elements-mat} (\text{mat-mod } M) = \text{elements-mat } M$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-vec-mod-row*:  $i < \text{dim-row } A \implies \text{row} (\text{mat-mod } A) i = \text{vec-mod} (\text{row } A i)$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-vec-mod-col*:  $j < \text{dim-col } A \implies \text{col} (\text{mat-mod } A) j = \text{vec-mod} (\text{col } A j)$   
 $\langle \text{proof} \rangle$

**lemma** *count-vec-mod-eq*:  $\text{set}_v v \subseteq \{0..<m\} \implies \text{count-vec } v x = \text{count-vec} (\text{vec-mod } v) x$   
 $\langle \text{proof} \rangle$

**lemma** *elems-mat-setv-row-0m*:  $i < \text{dim-row } M \implies \text{elements-mat } M \subseteq \{0..<m\} \implies \text{set}_v (\text{row } M i) \subseteq \{0..<m\}$   
 $\langle \text{proof} \rangle$

**lemma** *elems-mat-setv-col-0m*:  $j < \text{dim-col } M \implies \text{elements-mat } M \subseteq \{0..<m\} \implies \text{set}_v (\text{col } M j) \subseteq \{0..<m\}$   
 $\langle \text{proof} \rangle$

**lemma** *mat-mod-count-row-eq*:  $i < \text{dim-row } M \implies \text{elements-mat } M \subseteq \{0..<m\}$

```

 $\implies$ 
  count-vec (row (mat-mod M) i) x = count-vec (row M i) x
  ⟨proof⟩

lemma mat-mod-count-col-eq:  $j < \text{dim-col } M \implies \text{elements-mat } M \subseteq \{0..<m\}$ 
 $\implies$ 
  count-vec (col (mat-mod M) j) x = count-vec (col M j) x
  ⟨proof⟩

lemma mod-mat-one: mat-mod ( $1_m n$ ) = ( $1_m n$ )
  ⟨proof⟩

lemma mod-mat-zero: mat-mod ( $0_m nr nc$ ) = ( $0_m nr nc$ )
  ⟨proof⟩

lemma vec-mod-unit: vec-mod (unit-vec n i) = (unit-vec n i)
  ⟨proof⟩

lemma vec-mod-zero: vec-mod ( $0_v n$ ) = ( $0_v n$ )
  ⟨proof⟩

lemma mat-mod-cond-iff: elements-mat M  $\subseteq \{0..<m\} \implies P M \longleftrightarrow P (\text{mat-mod } M)$ 
  ⟨proof⟩

end

```

## 9.2 Mod Type

The below locale takes lemmas from the Poly Mod Finite Field theory in the Berlekamp Zassenhaus AFP entry, however has removed any excess material on polynomials mod, and only included the general factors. Ideally, this could be used as the base locale for both in the future

```

locale mod-type =
  fixes m :: int and ty :: 'a :: nontriv itself
  assumes m: m = CARD('a)
begin

lemma m1: m > 1 ⟨proof⟩

definition M :: int  $\Rightarrow$  int where M x = x mod m

lemma M-0[simp]: M 0 = 0
  ⟨proof⟩

lemma M-M[simp]: M (M x) = M x
  ⟨proof⟩

lemma M-plus[simp]: M (M x + y) = M (x + y) M (x + M y) = M (x + y)

```

$\langle proof \rangle$

**lemma**  $M\text{-minus}[simp]$ :  $M(Mx - y) = M(x - y)$   $M(x - My) = M(x - y)$   
 $\langle proof \rangle$

**lemma**  $M\text{-times}[simp]$ :  $M(Mx * y) = M(x * y)$   $M(x * My) = M(x * y)$   
 $\langle proof \rangle$

**lemma**  $M\text{-1}[simp]$ :  $M1 = 1$   $\langle proof \rangle$

**lemma**  $M\text{-sum}$ :  $M(\text{sum}(\lambda x. M(fx)) A) = M(\text{sum } f A)$   
 $\langle proof \rangle$

**definition**  $inv\text{-}M :: int \Rightarrow int$  **where**  
 $inv\text{-}M x = (\text{if } x + x \leq m \text{ then } x \text{ else } x - m)$

**lemma**  $M\text{-inv}\text{-}M\text{-id}[simp]$ :  $M(inv\text{-}M x) = Mx$   
 $\langle proof \rangle$

**definition**  $M\text{-Rel} :: int \Rightarrow 'a \text{ mod-ring} \Rightarrow \text{bool}$   
**where**  $M\text{-Rel } x x' \equiv (Mx = \text{to-int-mod-ring } x')$

**lemma**  $to\text{-int}\text{-mod-ring}\text{-plus}$ :  $to\text{-int}\text{-mod-ring}((x :: 'a \text{ mod-ring}) + y) = M(to\text{-int}\text{-mod-ring} x + to\text{-int}\text{-mod-ring} y)$   
 $\langle proof \rangle$

**lemma**  $to\text{-int}\text{-mod-ring}\text{-times}$ :  $to\text{-int}\text{-mod-ring}((x :: 'a \text{ mod-ring}) * y) = M(to\text{-int}\text{-mod-ring} x * to\text{-int}\text{-mod-ring} y)$   
 $\langle proof \rangle$

**lemma**  $eq\text{-}M\text{-Rel}[transfer\text{-rule}]$ :  $(M\text{-Rel} ==> M\text{-Rel} ==> (=)) (\lambda xy. Mx = My) (=)$   
 $\langle proof \rangle$

**lemma**  $one\text{-}M\text{-Rel}[transfer\text{-rule}]$ :  $M\text{-Rel } 1 1$   
 $\langle proof \rangle$

**lemma**  $zero\text{-}M\text{-Rel}[transfer\text{-rule}]$ :  $M\text{-Rel } 0 0$   
 $\langle proof \rangle$

**lemma**  $M\text{-to}\text{-int}\text{-mod-ring}$ :  $M(to\text{-int}\text{-mod-ring}(x :: 'a \text{ mod-ring})) = to\text{-int}\text{-mod-ring} x$   
 $\langle proof \rangle$

**lemma**  $right\text{-total}\text{-}M\text{-Rel}[transfer\text{-rule}]$ :  $right\text{-total } M\text{-Rel}$   
 $\langle proof \rangle$

**lemma**  $left\text{-total}\text{-}M\text{-Rel}[transfer\text{-rule}]$ :  $left\text{-total } M\text{-Rel}$

```

⟨proof⟩

lemma bi-total-M-Rel[transfer-rule]: bi-total M-Rel
⟨proof⟩

lemma to-int-mod-ring-of-int-M: to-int-mod-ring (of-int x :: 'a mod-ring) = M x
⟨proof⟩

lemma UNIV-M-Rel[transfer-rule]: rel-set M-Rel {0..<m} UNIV
⟨proof⟩

end

```

### 9.3 Mat mod type

Define a context to work on matrices and vectors of type '*a mod-ring*

```

locale mat-mod-type = mat-mod + mod-type
begin

```

```

lemma to-int-mod-ring-plus: to-int-mod-ring ((x :: 'a mod-ring) + y) = (to-int-mod-ring
x + to-int-mod-ring y) mod m
⟨proof⟩

```

```

lemma to-int-mod-ring-times: to-int-mod-ring ((x :: 'a mod-ring) * y) = (to-int-mod-ring
x * to-int-mod-ring y) mod m
⟨proof⟩

```

Set up transfer relation for matrices and vectors

```

definition MM-Rel :: int mat ⇒ 'a mod-ring mat ⇒ bool
where MM-Rel f f' ≡ (mat-mod f = to-int-mat f')

```

```

definition MV-Rel :: int vec ⇒ 'a mod-ring vec ⇒ bool
where MV-Rel v v' ≡ (vec-mod v = to-int-vec v')

```

```

lemma to-int-mat-index[simp]: i < dim-row N ⇒ j < dim-col N ⇒ (to-int-mat
N $$ (i, j)) = to-int-mod-ring (N $$ (i, j))
⟨proof⟩

```

```

lemma to-int-vec-index[simp]: i < dim-vec v ⇒ (to-int-vec v $i) = to-int-mod-ring
(v $i)
⟨proof⟩

```

```

lemma eq-dim-row-MM-Rel[transfer-rule]: (MM-Rel ==> (=)) dim-row dim-row
⟨proof⟩

```

```

lemma lt-dim-row-MM-Rel[transfer-rule]: (MM-Rel ==> (=) ==> (=)) (λ M
i. i < dim-row M) (λ M i. i < dim-row M)
⟨proof⟩

```

**lemma** *eq-dim-col-MM-Rel[transfer-rule]*:  $(MM\text{-}Rel \implies (=)) \ dim\text{-}col \ dim\text{-}col$   
 $\langle proof \rangle$

**lemma** *lt-dim-col-MM-Rel[transfer-rule]*:  $(MM\text{-}Rel \implies (=) \implies (=)) (\lambda M j. j < dim\text{-}col M) (\lambda M j. j < dim\text{-}col M)$   
 $\langle proof \rangle$

**lemma** *eq-dim-vec-MV-Rel[transfer-rule]*:  $(MV\text{-}Rel \implies (=)) \ dim\text{-}vec \ dim\text{-}vec$   
 $\langle proof \rangle$

**lemma** *lt-dim-vec-MV-Rel[transfer-rule]*:  $(MV\text{-}Rel \implies (=) \implies (=)) (\lambda v j. j < dim\text{-}vec v) (\lambda v j. j < dim\text{-}vec v)$   
 $\langle proof \rangle$

**lemma** *eq-MM-Rel[transfer-rule]*:  $(MM\text{-}Rel \implies MM\text{-}Rel \implies (=)) (\lambda f f'. mat\text{-}mod f = mat\text{-}mod f') (=)$   
 $\langle proof \rangle$

**lemma** *eq-MV-Rel[transfer-rule]*:  $(MV\text{-}Rel \implies MV\text{-}Rel \implies (=)) (\lambda v v'. vec\text{-}mod v = vec\text{-}mod v') (=)$   
 $\langle proof \rangle$

**lemma** *index-MV-Rel[transfer-rule]*:  $(MV\text{-}Rel \implies (=) \implies M\text{-}Rel)$   
 $(\lambda v i. if i < dim\text{-}vec v then v \$ i else 0) (\lambda v i. if i < dim\text{-}vec v then v \$ i else 0)$   
 $\langle proof \rangle$

**lemma** *index-MM-Rel[transfer-rule]*:  $(MM\text{-}Rel \implies (=) \implies (=) \implies M\text{-}Rel)$   
 $(\lambda M i j. if (i < dim\text{-}row M \wedge j < dim\text{-}col M) then M \$\$ (i, j) else 0)$   
 $(\lambda M i j. if (i < dim\text{-}row M \wedge j < dim\text{-}col M) then M \$\$ (i, j) else 0)$   
 $\langle proof \rangle$

**lemma** *index-MM-Rel-explicit*:  
**assumes**  $MM\text{-}Rel N N'$   
**assumes**  $i < dim\text{-}row N \ j < dim\text{-}col N$   
**shows**  $(N \$\$ (i, j)) \ mod \ m = to\text{-}int\text{-}mod\text{-}ring (N' \$\$ (i, j))$   
 $\langle proof \rangle$

**lemma** *one-MV-Rel[transfer-rule]*:  $MV\text{-}Rel (unit\text{-}vec n i) (unit\text{-}vec n i)$   
 $\langle proof \rangle$

**lemma** *one-MM-Rel[transfer-rule]*:  $MM\text{-}Rel (1_m n) (1_m n)$   
 $\langle proof \rangle$

**lemma** *zero-MM-Rel[transfer-rule]*:  $MM\text{-}Rel (0_m nr nc) (0_m nr nc)$   
 $\langle proof \rangle$

**lemma** *zero-MV-Rel[transfer-rule]*: *MV-Rel* ( $\theta_v$   $n$ ) ( $\theta_v$   $n$ )  
 *$\langle proof \rangle$*

**lemma** *right-unique-MV-Rel[transfer-rule]*: *right-unique MV-Rel*  
 *$\langle proof \rangle$*

**lemma** *right-unique-MM-Rel[transfer-rule]*: *right-unique MM-Rel*  
 *$\langle proof \rangle$*

**lemma** *mod-to-int-mod-ring*: *(to-int-mod-ring* ( $x :: 'a mod-ring$ ) *) mod m = to-int-mod-ring*  
 $x$   
 *$\langle proof \rangle$*

**lemma** *mat-mod-to-int-mat*: *mat-mod* (*to-int-mat* ( $N :: 'a mod-ring mat$ )) = *to-int-mat*  
 $N$   
 *$\langle proof \rangle$*

**lemma** *vec-mod-to-int-vec*: *vec-mod* (*to-int-vec* ( $v :: 'a mod-ring vec$ )) = *to-int-vec*  
 $v$   
 *$\langle proof \rangle$*

**lemma** *right-total-MM-Rel[transfer-rule]*: *right-total MM-Rel*  
 *$\langle proof \rangle$*

**lemma** *right-total-MV-Rel[transfer-rule]*: *right-total MV-Rel*  
 *$\langle proof \rangle$*

**lemma** *to-int-mod-ring-of-int-mod*: *to-int-mod-ring* (*of-int*  $x :: 'a mod-ring$ ) =  $x$   
*mod m*  
 *$\langle proof \rangle$*

**lemma** *vec-mod-v-representative*: *vec-mod v = to-int-vec* (*map-vec* *of-int v :: 'a mod-ring vec*)  
 *$\langle proof \rangle$*

**lemma** *mat-mod-N-representative*: *mat-mod N = to-int-mat* (*map-mat* *of-int N :: 'a mod-ring mat*)  
 *$\langle proof \rangle$*

**lemma** *left-total-MV-Rel[transfer-rule]*: *left-total MV-Rel*  
 *$\langle proof \rangle$*

**lemma** *left-total-MM-Rel[transfer-rule]*: *left-total MM-Rel*  
 *$\langle proof \rangle$*

**lemma** *bi-total-MV-Rel[transfer-rule]*: *bi-total MV-Rel*  
 *$\langle proof \rangle$*

**lemma** *bi-total-MM-Rel[transfer-rule]*: *bi-total MM-Rel*  
*(proof)*

**lemma** *domain-MV-rel[transfer-domain-rule]*: *Domain<sub>in</sub>p MV-Rel = (λ f. True)*  
*(proof)*

**lemma** *domain-MM-rel[transfer-domain-rule]*: *Domain<sub>in</sub>p MM-Rel = (λ f. True)*  
*(proof)*

**lemma** *mem-MV-Rel[transfer-rule]*:  
 $(MV\text{-}Rel \implies rel\text{-}set MV\text{-}Rel \implies (=)) (\lambda x Y. \exists y \in Y. vec\text{-}mod x = vec\text{-}mod y) (\in)$   
*(proof)*

**lemma** *mem-MM-Rel[transfer-rule]*:  
 $(MM\text{-}Rel \implies rel\text{-}set MM\text{-}Rel \implies (=)) (\lambda x Y. \exists y \in Y. mat\text{-}mod x = mat\text{-}mod y) (\in)$   
*(proof)*

**lemma** *conversep-MM-Rel-OO-MM-Rel [simp]*:  $MM\text{-}Rel^{-1-1} OO MM\text{-}Rel = (=)$   
*(proof)*

**lemma** *MM-Rel-OO-conversep-MM-Rel [simp]*:  $MM\text{-}Rel OO MM\text{-}Rel^{-1-1} = (\lambda M M'. mat\text{-}mod M = mat\text{-}mod M')$   
*(proof)*

**lemma** *conversep-MM-Rel-OO-eq-m [simp]*:  $MM\text{-}Rel^{-1-1} OO (\lambda M M'. mat\text{-}mod M = mat\text{-}mod M') = MM\text{-}Rel^{-1-1}$   
*(proof)*

**lemma** *eq-m-OO-MM-Rel [simp]*:  $(\lambda M M'. mat\text{-}mod M = mat\text{-}mod M') OO MM\text{-}Rel = MM\text{-}Rel$   
*(proof)*

**lemma** *eq-mset-MM-Rel [transfer-rule]*:  
 $(rel\text{-}mset MM\text{-}Rel \implies rel\text{-}mset MM\text{-}Rel \implies (=)) (rel\text{-}mset (\lambda M M'. mat\text{-}mod M = mat\text{-}mod M')) (=)$   
*(proof)*

**lemma** *vec-mset-MV-Rel[transfer-rule]*:  
 $(MV\text{-}Rel \implies (=)) (\lambda v. vec\text{-}mset (vec\text{-}mod v)) (\lambda v. image\text{-}mset (to\text{-}int\text{-}mod\text{-}ring) (vec\text{-}mset v))$   
*(proof)*

**lemma** *vec-count-MV-Rel-direct*:  
**assumes** *MV-Rel v1 v2*  
**shows** *count-vec v2 i = count-vec (vec-mod v1) (to-int-mod-ring i)*  
*(proof)*

```

lemma MM-Rel-MV-Rel-row: MM-Rel A B  $\implies$  i < dim-row A  $\implies$  MV-Rel (row A i) (row B i)
  ⟨proof⟩

lemma MM-Rel-MV-Rel-col: MM-Rel A B  $\implies$  j < dim-col A  $\implies$  MV-Rel (col A j) (col B j)
  ⟨proof⟩

end
end

```

## 10 Variations on Fisher's Inequality

```

theory Fishers-Inequality-Variations imports Dual-Systems Rank-Argument-General
Vector-Matrix-Mod Linear-Bound-Argument
begin

```

### 10.1 Matrix mod properties

This is reasoning on properties specific to incidence matrices under *mat-mod*. Ultimately, this definition was not used in the final proof but it is left as is in case of future use

```

context mat-mod
begin

```

```

lemma mat-mod-proper-iff: proper-inc-mat (mat-mod N)  $\longleftrightarrow$  proper-inc-mat N
  ⟨proof⟩

```

```

lemma mat-mod-rep-num-eq: i < dim-row N  $\implies$  elements-mat N  $\subseteq \{0..<m\}$ 
 $\implies$ 
  mat-rep-num (mat-mod N) i = mat-rep-num N i
  ⟨proof⟩

```

```

lemma mat-point-index-eq: elements-mat N  $\subseteq \{0..<m\} \implies$ 
  mat-point-index (mat-mod N) I = mat-point-index N I
  ⟨proof⟩

```

```

lemma mod-mat-inter-num-eq: elements-mat N  $\subseteq \{0..<m\} \implies$ 
  mat-inter-num (mat-mod N) j1 j2 = mat-inter-num N j1 j2
  ⟨proof⟩

```

```

lemma mod-mat-block-size: elements-mat N  $\subseteq \{0..<m\} \implies$  mat-block-size (mat-mod N) j = mat-block-size N j
  ⟨proof⟩

```

```

lemma mat-mod-non-empty-col-iff: elements-mat M  $\subseteq \{0..<m\} \implies$ 
  non-empty-col (mat-mod M) j  $\longleftrightarrow$  non-empty-col M j
  ⟨proof⟩

```

```

end

context mat-mod-type
begin

lemma mat-rep-num-MM-Rel:
  assumes MM-Rel A B
  assumes i < dim-row A
  shows mat-rep-num (mat-mod A) i = mat-rep-num B i
  ⟨proof⟩

lemma mat-block-size-MM-Rel:
  assumes MM-Rel A B
  assumes j < dim-col A
  shows mat-block-size (mat-mod A) j = mat-block-size B j
  ⟨proof⟩

lemma mat-inter-num-MM-Rel:
  assumes MM-Rel A B
  assumes j1 < dim-col A j2 < dim-col B
  shows mat-inter-num (mat-mod A) j1 j2 = mat-inter-num B j1 j2
  ⟨proof⟩

  Lift 01 and mat mod equivalence on 0-1 matrices

lemma of-int-mod-ring-lift-01-eq:
  assumes zero-one-matrix N
  shows map-mat (of-int-mod-ring) N = (lift-01-mat) N
  ⟨proof⟩

lemma to-int-mod-ring-lift-01-eq:
  assumes zero-one-matrix N
  shows to-int-mat N = (lift-01-mat) N
  ⟨proof⟩

end

```

## 10.2 The Odd-town Problem

The odd-town problem [1] is perhaps one of the most common introductory problems for applying the linear algebra bound method to a combinatorial problem. In mathematical literature, it is considered simpler than Fisher's Inequality, however presents some interesting challenges to formalisation. Most significantly, it considers the incidence matrix to have elements of types integers mod 2.

Initially, define a locale to represent the odd town context (a town with  $v$  people, and  $b$  groups) which must each be of odd size, but have an even intersection number with any other group

```

locale odd-town = ordered-design +
  assumes odd-groups:  $bl \in \# \mathcal{B} \implies \text{odd}(\text{card } bl)$ 
  and even-inters:  $bl1 \in \# \mathcal{B} \implies bl2 \in \# (\mathcal{B} - \{\#bl1\}) \implies \text{even}(bl1 \cap bl2)$ 
begin

lemma odd-town-no-repeat-clubs: distinct-mset  $\mathcal{B}$ 
   $\langle \text{proof} \rangle$ 

lemma odd-blocks-mat-block-size:  $j < \text{dim-col } N \implies \text{odd}(\text{mat-block-size } N j)$ 
   $\langle \text{proof} \rangle$ 

lemma odd-block-size-mod-2:
  assumes  $\text{CARD}('b:\text{prime-card}) = 2$ 
  assumes  $j < b$ 
  shows of-nat  $(\text{card } (\mathcal{B}s ! j)) = (1 :: 'b \text{ mod-ring})$ 
   $\langle \text{proof} \rangle$ 

lemma valid-indices-block-min:  $j1 < \text{dim-col } N \implies j2 < \text{dim-col } N \implies j1 \neq j2$ 
   $\implies b \geq 2$ 
   $\langle \text{proof} \rangle$ 

lemma even-inter-mat-intersections:  $j1 < \text{dim-col } N \implies j2 < \text{dim-col } N \implies j1 \neq j2$ 
   $\implies \text{even}(\text{mat-inter-num } N j1 j2)$ 
   $\langle \text{proof} \rangle$ 

lemma even-inter-mod-2:
  assumes  $\text{CARD}('b:\text{prime-card}) = 2$ 
  assumes  $i < b$  and  $jlt: j < b$  and  $ne: i \neq j$ 
  shows of-nat  $((\mathcal{B}s ! i) \cap (\mathcal{B}s ! j)) = (0 :: 'b \text{ mod-ring})$ 
   $\langle \text{proof} \rangle$ 

end

```

The odd town locale must be simple by definition

```

sublocale odd-town  $\subseteq$  ordered-simple-design
   $\langle \text{proof} \rangle$ 

```

```

context odd-town
begin

```

The upper bound lemma (i.e. variation on Fisher's) for the odd town property using the linear bound argument. Notice it follows exactly the same pattern as the generalised version, however it's sum manipulation argument is significantly simpler (in line with the mathematical proofs)

```

lemma upper-bound-clubs:
  assumes  $\text{CARD}('b:\text{prime-card}) = 2$ 
  shows  $b \leq v$ 
   $\langle \text{proof} \rangle$ 

```

```

end

end

theory Fishers-Inequality-Root
imports
  Set-Multiset-Extras
  Matrix-Vector-Extras
  Design-Extras

  Incidence-Matrices
  Dual-Systems
  Rank-Argument-General
  Linear-Bound-Argument

  Fishers-Inequality
  Vector-Matrix-Mod
  Fishers-Inequality-Variations
begin

end

```

## References

- [1] L. Babai and P. Frankl. *Linear Algebra Methods in Combinatorics*. 2.1 edition, 2020.
- [2] B. Bukh. Lecture notes in algebraic Methods in Combinatorics: Rank argument, 2014.
- [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] R. A. Fisher. An Examination of the Different Possible Solutions of a Problem in Incomplete Blocks. *Annals of Eugenics*, 10(1):52–75, 1940.
- [5] C. D. Godsil. Tools from Linear Algebra. In L. L. Graham RL, Grötschel M, editor, *Handbook of Combinatorics*, volume 2. Elsevier, Amsterdam.
- [6] S. Herke. Math3301 lecture notes in combinatorial design theory, July 2016.
- [7] S. Jukna. *Extremal Combinatorics*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

- [8] D. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer, 2004.