

cheat sheet finset.v (SSREFLECT v1.15)

ssrfun.v naming conventions

K cancel
 LR move an op from the lhs of a rel to the rhs
 RL move an op from the rhs to the lhs

ssrfun.v definitions

```

injective f      forall x1 x2, f x1 = f x2 -> x1 = x2
cancel f g      g (f x) = x
involutive f    cancel f f
left_injective op injective (op-1 x)
right_injective op injective (op y)
left_id e op    e □ x = x
right_id e op  x □ e = x
left_zero z op z □ x = z
right_zero z op x □ z = z
self_inverse e op x □ x = e
idempotent op  x □ x = x
commutative op x □ y = y □ x
associative op x □ (y □ z) = (x □ y) □ z
right_commutative op (x □ y) □ z = (x □ z) □ y
left_commutative op x □ (y □ z) = y □ (x □ z)
left_distributive op add (x + y) * z = (x * z) + (y * z)
right_distributive op add x * (y + z) = (x * y) + (x * z)
left_loop inv op cancel (op x) (op (inv x))
    
```

ssrbool.v naming conventions

A associativity
 AC right commutativity
 b a boolean argument
 C commutativity/complement
 D predicate difference
 E elimination
 F/f boolean false
 T/t boolean truth
 U predicate union

finset.v naming conventions

∅ the empty set
 T the full set
 1 singleton set
 C complement
 U union
 I intersection
 D difference

(* set_scope *)

```

× set0
A :|: B setU
a |: A [set a] :|: A
A :&: B setI
~: A setC A
A :\: B setD A B
A :\ a A :\: [set a]
f @~-1: A preimset f (mem A)
f @: A imset f (mem A)
f @2: ( A , B ) imset2 f (mem A) (fun _ =>mem B) f(A, B)
    
```

(* bool_scope *)

```

∅
a \in A see ssrbool.v a ∈ A
A \subset B see fintype.v A ⊆ B
[disjoint A & B] see fintype.v A ∩ B = ∅
    
```

```

setP A =i B <-> A = B
in_set0 x \in set0 = false
subset0 (A \subset set0) = (A == set0)
in_set1 (x \in [set a]) = (x == a)
in_setD1 (x \in A :\ b) = (x != b) && (x \in A)
in_setU (x \in A :|: B) = (x \in A) || (x \in B)
in_setC (x \in ~: A) = (x \notin A)
(NB: inE is a multi-rule corresponding to in_set0, in_set1, in_setD1, in_setU, in_setC, etc.)
setUC A :|: B = B :|: A
setIC A :&: B = B :&: A
setKI A :|: (B :&: A) = A
setCI ~: (A :&: B) = ~: A :|: ~: B
setCK involutive (@setC T)
setD0 A :\: set0 =A
cardsE #|[set x in pA]| = #|pA| (NB: cardE : #|A| = size (enum A) in fintype.v)
cards0 #|@set0 T| = 0 (NB: card0 : #|@pred0 T|=0 in fintype.v)
cards_eq0 (#|A| == 0) = (A == set0)
cardsU #|A :|: B| = #|A| + #|B| - #|A :&: B|
cardsT #|[set: T]| = #|T| (NB: cardT : #|T| = size (enum T) in fintype.v)
set0Pn reflect (exists x, x \in A) (A != set0)
subsetI1 A :&: B \subset A
subsetUr B \subset A :|: B
subsetI (A \subset B :&: C) = (A \subset B) && (A \subset C)
setI_eq0 (A :&: B == set0) = [disjoint A & B]
imsetP reflect (exists2 x, in_mem x D & y = f x) (y \in imset f D)
card_imset injective f -> #|f @: D| = #|D|
    
```

Section Partitions

cover P $\bigcup_{B \in P} B$
 trivIset P $\sum_{B \in P} |B| = |\text{cover}(P)|$
 see also bigop_doc.pdf