cheat sheet `finset.v` (SSReflect v1.5)

`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^~ x)`
`right_injective op`	`injective (op y)`
`left_id e op`	`op e x = x`
`right_id e op`	`op x e = x`
`left_zero z op`	`op z x = z`
`right_commutative op`	`op (op x y) z = op (op x z) y`
`right_zero z op`	`op x z = z`
`left_commutative op`	`op x (op y z) = op y (op x z)`
`left_distributive op add`	`op (add x y) z = add (op x z) (op y z)`
`right_distributive op add`	`op x (add y z) = add (op x y) (op x z)`
`left_loop inv op`	`cancel (op x) (op (inv x))`
`self_inverse e op`	`op x x = e`
`commutative op`	`op x y = op y x`
`idempotent op`	`op x x = x`
`associative op`	`op x (op y z) = op (op x y) z`

`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

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

`set_scope`

<table>
<tr><td>$\times$</td><td><code>set0</code></td><td>$\emptyset$</td></tr>
<tr><td><code>A :|: B</code></td><td><code>setU</code></td><td>$A \cup B$</td></tr>
<tr><td><code>a |: A</code></td><td><code>[set a] :|: A</code></td><td>$\{a\} \cup A$</td></tr>
<tr><td><code>A :&: B</code></td><td><code>setI</code></td><td>$A \cap B$</td></tr>
<tr><td><code>~: A</code></td><td><code>setC A</code></td><td>A^C</td></tr>
<tr><td><code>A :\: B</code></td><td><code>setD A B</code></td><td>$A \backslash B$</td></tr>
<tr><td><code>A :\ a</code></td><td><code>A :\: [set a]</code></td><td>$A \backslash \{a\}$</td></tr>
<tr><td><code>f @^-1: A</code></td><td><code>preimset f (mem A)</code></td><td>$f^{-1}(A)$</td></tr>
<tr><td><code>f @: A</code></td><td><code>imset f (mem A)</code></td><td>$f(A)$</td></tr>
<tr><td><code>f @2: (A , B)</code></td><td><code>imset2 f (mem A) (fun _ =>mem B)</code></td><td>$f(A, B)$</td></tr>
</table>

`bool_scope`

<table>
<tr><td><code>a \in A</code></td><td>see <code>ssrbool.v</code></td><td>$a \in A$</td></tr>
<tr><td><code>A \subset B</code></td><td>see <code>fintype.v</code></td><td>$A \subseteq B$</td></tr>
<tr><td><code>[disjoint A & B]</code></td><td>see <code>fintype.v</code></td><td>$A \cap B = \emptyset$</td></tr>
</table>

<table>
<tr><td><code>setP</code></td><td><code>A =i B <-> A = B</code></td></tr>
<tr><td><code>in_set0</code></td><td><code>x \in set0 = false</code></td></tr>
<tr><td><code>subset0</code></td><td><code>(A \subset set0) = (A == set0)</code></td></tr>
<tr><td><code>in_set1</code></td><td><code>(x \in [set a]) = (x == a)</code></td></tr>
<tr><td><code>in_setD1</code></td><td><code>(x \in A :\ b) = (x != b) && (x \in A)</code></td></tr>
<tr><td><code>in_setU</code></td><td><code>(x \in A :|: B) = (x \in A) || (x \in B)</code></td></tr>
<tr><td><code>in_setC</code></td><td><code>(x \in ~: A) = (x \notin A)</code></td></tr>
<tr><td colspan="2">(NB: <code>inE</code>: <code>in_set0</code>, <code>in_set1</code>, <code>in_setD1</code>, <code>in_setU</code>, <code>in_setC</code>, etc.)</td></tr>
<tr><td><code>setUC</code></td><td><code>A :|: B = B :|: A</code></td></tr>
<tr><td><code>setIC</code></td><td><code>A :&: B = B :&: A</code></td></tr>
<tr><td><code>setKI</code></td><td><code>A :|: (B :&: A) = A</code></td></tr>
<tr><td><code>setCI</code></td><td><code>~: (A :&: B) = ~: A :|: ~: B</code></td></tr>
<tr><td><code>setCK</code></td><td><code>involutive (@setC T)</code></td></tr>
<tr><td><code>setD0</code></td><td><code>A :\: set0 =A</code></td></tr>
<tr><td><code>cardsE</code></td><td><code>#|[set x in pA]| = #|pA|</code> (NB: <code>cardE : #|A|= size (enum A)</code> in <code>fintype.v</code>)</td></tr>
<tr><td><code>cards0</code></td><td><code>#|@set0 T| = 0</code> (NB: <code>card0 : #|@pred0 T|=0</code> in <code>fintype.v</code>)</td></tr>
<tr><td><code>cards_eq0</code></td><td><code>(#|A| == 0) = (A == set0)</code></td></tr>
<tr><td><code>cardsU</code></td><td><code>#|A :|: B| = #|A| + #|B| - #|A :&: B|</code></td></tr>
<tr><td><code>cardsT</code></td><td><code>#|[set: T]| = #|T|</code> (NB: <code>cardT : #|T|= size (enum T)</code> in <code>fintype.v</code>)</td></tr>
<tr><td><code>set0Pn</code></td><td><code>reflect (exists x, x \in A) (A != set0)</code></td></tr>
<tr><td><code>subsetIl</code></td><td><code>A :&: B \subset A</code></td></tr>
<tr><td><code>subsetUr</code></td><td><code>B \subset A :|: B</code></td></tr>
<tr><td><code>subsetI</code></td><td><code>(A \subset B :&: C) = (A \subset B) && (A \subset C)</code></td></tr>
<tr><td><code>setI_eq0</code></td><td><code>(A :&: B == set0) = [disjoint A & B]</code></td></tr>
<tr><td><code>imsetP</code></td><td><code>reflect (exists2 x, in_mem x D & y = f x) (y \in imset f D)</code></td></tr>
<tr><td><code>card_imset</code></td><td><code>injective f ->#|f @: D|=#|D|</code></td></tr>
</table>

Section Partitions

`cover P`	$\bigcup_{B \in P} B$
`trivIset P`	$\sum_{B \in P} \lvert B \rvert = \lvert cover(P) \rvert$

see also `bigop_doc.pdf`