

# Finite Map Extras

Javier Díaz  
<javier.diaz.manzi@gmail.com>

May 26, 2024

## Abstract

This includes useful syntactic sugar, new operators and functions and their associated lemmas for finite maps which currently are not present in the standard `Finite_Map` theory.

## Contents

### 1 Extra Features for Finite Maps

1

## 1 Extra Features for Finite Maps

**theory** *Finite-Map-Extras*

**imports** *HOL-Library.Finite-Map*

**begin**

Extra lemmas and syntactic sugar for *fmap*

**notation** *fmlookup* (**infixl**  $\langle\$\$\rangle$  900)

**notation** *fmempty* ( $\langle\{\$\$\}\rangle$ )

**nonterminal** *fmaplets* and *fmaplet*

**syntax**

*-fmaplet* :: [*'a*, *'a*]  $\Rightarrow$  *fmaplet* (- / $\$\$$ :=/ -)  
*-fmaplets* :: [*'a*, *'a*]  $\Rightarrow$  *fmaplet* (- / [ $\$\$$ :=] / -)  
          :: *fmaplet*  $\Rightarrow$  *fmaplets* (-)  
*-Fmaplets* :: [*fmaplet*, *fmaplets*]  $\Rightarrow$  *fmaplets* (-, / -)  
*-FmapUpd* :: [(*'a*, *'b*) *fmap*, *fmaplets*]  $\Rightarrow$  (*'a*, *'b*) *fmap* (-/'(-) [900, 0] 900)  
*-Fmap*    :: *fmaplets*  $\Rightarrow$  (*'a*, *'b*) *fmap* ((1{-}))

**translations**

*-FmapUpd* *m* (*-Fmaplets* *xy* *ms*)  $\equiv$  *-FmapUpd* (*-FmapUpd* *m* *xy*) *ms*  
*-FmapUpd* *m* (*-fmaplet* *x* *y*)  $\equiv$  *CONST* *fmupd* *x* *y* *m*  
*-Fmap* *ms*  $\equiv$  *-FmapUpd* (*CONST* *fmempty*) *ms*  
*-Fmap* (*-Fmaplets* *ms1* *ms2*)  $\leftarrow$  *-FmapUpd* (*-Fmap* *ms1*) *ms2*

$-Fmaplets\ ms1\ (-Fmaplets\ ms2\ ms3) \leftarrow -Fmaplets\ (-Fmaplets\ ms1\ ms2)\ ms3$

**abbreviation** *fmap-lookup-the* (**infixl**  $\langle \$\$ \rangle\ 900$ ) **where**  
 $m\ \$\$ k \equiv \text{the } (m\ \$\$ k)$

**lemma** *fmadd-singletons-comm*:

**assumes**  $k_1 \neq k_2$

**shows**  $\{k_1\ \$\$ := v_1\} ++_f \{k_2\ \$\$ := v_2\} = \{k_2\ \$\$ := v_2\} ++_f \{k_1\ \$\$ := v_1\}$

$\langle \text{proof} \rangle$

**lemma** *fmap-singleton-comm*:

**assumes**  $m\ \$\$ k = \text{None}$

**shows**  $m ++_f \{k\ \$\$ := v\} = \{k\ \$\$ := v\} ++_f m$

$\langle \text{proof} \rangle$

**lemma** *fmap-disj-comm*:

**assumes**  $fmdom'\ m_1 \cap fmdom'\ m_2 = \{\}$

**shows**  $m_1 ++_f m_2 = m_2 ++_f m_1$

$\langle \text{proof} \rangle$

**lemma** *fmran-singleton*:  $fmran\ \{k\ \$\$ := v\} = \{|v|\}$

$\langle \text{proof} \rangle$

**lemma** *fmmmap-keys-hom*:

**assumes**  $fmdom'\ m_1 \cap fmdom'\ m_2 = \{\}$

**shows**  $fmmmap\text{-keys}\ f\ (m_1 ++_f m_2) = fmmmap\text{-keys}\ f\ m_1 ++_f fmmmap\text{-keys}\ f\ m_2$

$\langle \text{proof} \rangle$

**lemma** *map-insort-is-insort-key*:

**assumes**  $m\ \$\$ k = \text{None}$

**shows**  $\text{map } (\lambda k'. (k', m(k\ \$\$ := v)\ \$\$ k')) (\text{insort } k\ xs) =$

$\text{insort-key } fst\ (k, v)\ (\text{map } (\lambda k'. (k', m(k\ \$\$ := v)\ \$\$ k'))\ xs)$

$\langle \text{proof} \rangle$

**lemma** *sorted-list-of-fmap-is-insort-key-fst*:

**assumes**  $m\ \$\$ k = \text{None}$

**shows**  $\text{sorted-list-of-fmap } (m(k\ \$\$ := v)) = \text{insort-key } fst\ (k, v)\ (\text{sorted-list-of-fmap } m)$

$\langle \text{proof} \rangle$

**lemma** *distinct-fst-inj*:

**assumes**  $\text{distinct } (\text{map } fst\ ps)$

**and**  $\text{inj } f$

**shows**  $\text{distinct } (\text{map } fst\ (\text{map } (\lambda(k, v). (f\ k, v))\ ps))$

$\langle \text{proof} \rangle$

**lemma** *distinct-sorted-list-of-fmap*:

**shows**  $\text{distinct } (\text{map } fst\ (\text{sorted-list-of-fmap } m))$

$\langle \text{proof} \rangle$

**lemma** *map-inj-pair-non-membership*:

**assumes**  $k \notin \text{set } (\text{map } \text{fst } ps)$   
**and** *inj*  $f$   
**shows**  $f k \notin \text{set } (\text{map } \text{fst } (\text{map } (\lambda(k, v). (f k, v)) ps))$   
*<proof>*

**lemma** *map-insort-key-fst*:

**assumes** *distinct*  $(\text{map } \text{fst } ps)$   
**and**  $k \notin \text{set } (\text{map } \text{fst } ps)$   
**and** *inj*  $f$   
**and** *mono*  $f$   
**shows**  $\text{map } (\lambda(k, v). (f k, v)) (\text{insort-key } \text{fst } (k, v) ps) =$   
 $\text{insort-key } \text{fst } (f k, v) (\text{map } (\lambda(k, v). (f k, v)) ps)$   
*<proof>*

**lemma** *map-sorted-list-of-fmap*:

**assumes** *inj*  $f$   
**and** *mono*  $f$   
**and**  $m \ \$\$ k = \text{None}$   
**shows**  $\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } (m(k \ \$\$ := v))) =$   
 $\text{insort-key } \text{fst } (f k, v) (\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } m))$   
*<proof>*

**lemma** *fmap-of-list-insort-key-fst*:

**assumes** *distinct*  $(\text{map } \text{fst } ps)$   
**and**  $k \notin \text{set } (\text{map } \text{fst } ps)$   
**shows**  $\text{fmap-of-list } (\text{insort-key } \text{fst } (k, v) ps) = (\text{fmap-of-list } ps)(k \ \$\$ := v)$   
*<proof>*

**lemma** *fmap-of-list-insort-key-fst-map*:

**assumes** *inj*  $f$   
**and**  $m \ \$\$ k = \text{None}$   
**shows**  $\text{fmap-of-list } (\text{insort-key } \text{fst } (f k, v) (\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } m))) =$   
 $(\text{fmap-of-list } (\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } m)))(f k \ \$\$ := v)$   
*<proof>*

**lemma** *fmap-of-list-sorted-list-of-fmap*:

**fixes**  $m :: ('a::\text{linorder}, 'b) \text{fmap}$   
**and**  $f :: 'a \Rightarrow 'c::\text{linorder}$   
**assumes** *inj*  $f$   
**and** *mono*  $f$   
**and**  $m \ \$\$ k = \text{None}$   
**shows**  $\text{fmap-of-list } (\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } (m(k \ \$\$ := v)))) =$   
 $(\text{fmap-of-list } (\text{map } (\lambda(k, v). (f k, v)) (\text{sorted-list-of-fmap } m)))(f k \ \$\$ := v)$   
*<proof>*

Map difference

**lemma** *fsubset-antisym*:

**assumes**  $m \subseteq_f n$

**and**  $n \subseteq_f m$   
**shows**  $m = n$   
 ⟨proof⟩

**abbreviation**

$fmdiff :: ('a, 'b) fmap \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$  (**infixl** <-- $_f$ > 100) **where**  
 $m_1 --_f m_2 \equiv fmfiter (\lambda x. x \notin fmdom' m_2) m_1$

**lemma** *fmdiff-partition*:

**assumes**  $m_2 \subseteq_f m_1$   
**shows**  $m_2 ++_f (m_1 --_f m_2) = m_1$   
 ⟨proof⟩

**lemma** *fmdiff-fmupd*:

**assumes**  $m \ \$\$ k = None$   
**shows**  $m(k \ \$\$ := v) --_f \{k \ \$\$ := v\} = m$   
 ⟨proof⟩

Map symmetric difference

**abbreviation** *fmsym-diff* ::  $('a, 'b) fmap \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$  (**infixl** < $\Delta_f$ > 100) **where**  
 $m_1 \Delta_f m_2 \equiv (m_1 --_f m_2) ++_f (m_2 --_f m_1)$

Domain restriction

**abbreviation** *dom-res* ::  $'a \text{ set} \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$  (**infixl** < $\triangleleft$ > 150) **where**  
 $s \triangleleft m \equiv fmfiter (\lambda x. x \in s) m$

Domain exclusion

**abbreviation** *dom-exc* ::  $'a \text{ set} \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$  (**infixl** < $\triangleleft'$ > 150) **where**  
 $s \triangleleft' m \equiv fmfiter (\lambda x. x \notin s) m$

Intersection plus

**abbreviation** *intersection-plus* ::  $('a, 'b::\text{monoid-add}) fmap \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$   
 (**infixl** < $\cap_+$ > 100)

**where**

$m_1 \cap_+ m_2 \equiv fmmmap-keys (\lambda k v. v + m_1 \ \$\$! k) (fmdom' m_1 \triangleleft m_2)$

Union override right

**abbreviation** *union-override-right* ::  $('a, 'b) fmap \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$   
 (**infixl** < $\cup_{\rightarrow}$ > 100)

**where**

$m_1 \cup_{\rightarrow} m_2 \equiv (fmdom' m_2 \triangleleft' m_1) ++_f m_2$

Union override left

**abbreviation** *union-override-left* ::  $('a, 'b) fmap \Rightarrow ('a, 'b) fmap \Rightarrow ('a, 'b) fmap$   
 (**infixl** < $\cup_{\leftarrow}$ > 100)

**where**

$m_1 \cup_{\leftarrow} m_2 \equiv m_1 ++_f (fmdom' m_1 \triangleleft' m_2)$

Union override plus

**abbreviation** *union-override-plus* :: ('a, 'b::monoid-add) fmap ⇒ ('a, 'b) fmap ⇒ ('a, 'b) fmap  
(**infixl** <math>\cup\_+</math> 100)

**where**

$$m_1 \cup_+ m_2 \equiv (m_1 \Delta_f m_2) ++_f (m_1 \cap_+ m_2)$$

Extra lemmas for the non-standard map operators

**lemma** *dom-res-singleton*:

**assumes**  $m \ \$\$ \ k = \text{Some } v$

**shows**  $\{k\} \triangleleft m = \{k \ \$\$ := v\}$

*<proof>*

**lemma** *dom-res-union-distr*:

**shows**  $(A \cup B) \triangleleft m = A \triangleleft m ++_f B \triangleleft m$

*<proof>*

**lemma** *dom-exc-add-distr*:

**shows**  $s \triangleleft / (m_1 ++_f m_2) = (s \triangleleft / m_1) ++_f (s \triangleleft / m_2)$

*<proof>*

**lemma** *fmap-partition*:

**shows**  $m = s \triangleleft / m ++_f s \triangleleft m$

*<proof>*

**lemma** *dom-res-addition-in*:

**assumes**  $m_1 \ \$\$ \ k = \text{None}$

**and**  $m_2 \ \$\$ \ k = \text{Some } v'$

**shows**  $\text{fndom}' (m_1(k \ \$\$ := v)) \triangleleft m_2 = \text{fndom}' m_1 \triangleleft m_2 ++_f \{k \ \$\$ := v'\}$

*<proof>*

**lemma** *dom-res-addition-not-in*:

**assumes**  $m_2 \ \$\$ \ k = \text{None}$

**shows**  $\text{fndom}' (m_1(k \ \$\$ := v)) \triangleleft m_2 = \text{fndom}' m_1 \triangleleft m_2$

*<proof>*

**lemma** *inter-plus-addition-in*:

**assumes**  $m_1 \ \$\$ \ k = \text{None}$

**and**  $m_2 \ \$\$ \ k = \text{Some } v'$

**shows**  $m_1(k \ \$\$ := v) \cap_+ m_2 = (m_1 \cap_+ m_2) ++_f \{k \ \$\$ := v' + v\}$

*<proof>*

**lemma** *inter-plus-addition-notin*:

**assumes**  $m_1 \ \$\$ \ k = \text{None}$

**and**  $m_2 \ \$\$ \ k = \text{None}$

**shows**  $m_1(k \ \$\$ := v) \cap_+ m_2 = (m_1 \cap_+ m_2)$

*<proof>*

**lemma** *union-plus-addition-notin*:

**assumes**  $m_1 \ \$\$ \ k = \text{None}$

**and**  $m_2 \text{ } k = \textit{None}$   
**shows**  $m_1(k \text{ } := v) \cup_+ m_2 = (m_1 \cup_+ m_2) ++_f \{k \text{ } := v\}$   
*<proof>*  
**end**