Finite Map Extras

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

March 17, 2025

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

theory *Finite-Map-Extras*
  imports *HOL−Library.Finite-Map*
begin

Extra lemmas and syntactic sugar for *fmap*

notation *fmlookup* (**infixl** ‹$$› *900*)

notation *fmempty* (‹{$$}›)

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

syntax
  -fmaplet :: ['*a*, '*a*] ⇒ *fmaplet* (‹- /$$:=/ ->)
  -fmaplets :: ['*a*, '*a*] ⇒ *fmaplet* (‹- /[$$:=]/ ->)
    :: *fmaplet* ⇒ *fmaplets* (‹->)
  -Fmaplets :: [*fmaplet*, *fmaplets*] ⇒ *fmaplets* (‹-,/ ->)
  -FmapUpd :: [('*a*, '*b*) *fmap*, *fmaplets*] ⇒ ('*a*, '*b*) *fmap* (‹-/'(-')› [*900*, *0*] *900*)
  -Fmap :: *fmaplets* ⇒ ('*a*, '*b*) *fmap* (‹(*1*{-})›)

syntax-consts
  -fmaplet -fmaplets -Fmaplets -FmapUpd -Fmap ⇌ *fmupd*

translations
  -FmapUpd m (*-Fmaplets xy ms*) ⇌ *-FmapUpd* (*-FmapUpd m xy*) *ms*

-FmapUpd m (-fmaplet x y) $\rightleftharpoons$ *CONST fmupd x y m*
-Fmap ms $\rightleftharpoons$ *-FmapUpd (CONST fmempty) ms*
-Fmap (-Fmaplets ms1 ms2) $\leftharpoonup$ *-FmapUpd (-Fmap ms1) ms2*
-Fmaplets ms1 (-Fmaplets ms2 ms3) $\leftharpoonup$ *-Fmaplets (-Fmaplets ms1 ms2) ms3*

abbreviation *fmap-lookup-the* (**infixl** ‹$$!› *900*) **where**
m $$! $k \equiv$ *the* (m $$ k)

lemma *fmadd-singletons-comm*:
assumes $k_1 \neq k_2$
shows $\{k_1$ $$:= $v_1\} +\!\!+_f \{k_2$ $$:= $v_2\} = \{k_2$ $$:= $v_2\} +\!\!+_f \{k_1$ $$:= $v_1\}$
⟨*proof*⟩

lemma *fmap-singleton-comm*:
assumes m $$ k = *None*
shows $m +\!\!+_f \{k$ $$:= $v\} = \{k$ $$:= $v\} +\!\!+_f m$
⟨*proof*⟩

lemma *fmap-disj-comm*:
assumes *fmdom'* $m_1 \cap$ *fmdom'* $m_2 = \{\}$
shows $m_1 +\!\!+_f m_2 = m_2 +\!\!+_f m_1$
⟨*proof*⟩

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

lemma *fmmap-keys-hom*:
assumes *fmdom'* $m_1 \cap$ *fmdom'* $m_2 = \{\}$
shows *fmmap-keys f* ($m_1 +\!\!+_f m_2$) = *fmmap-keys f* $m_1 +\!\!+_f$ *fmmap-keys f* m_2
⟨*proof*⟩

lemma *map-insort-is-insort-key*:
assumes m $$ k = *None*
shows *map* ($\lambda k'$. (k', $m(k$ $$:= $v)$ $$! k')) (*insort k xs*) =
insort-key fst (k, v) (*map* ($\lambda k'$. (k', $m(k$ $$:= $v)$ $$! k')) *xs*)
⟨*proof*⟩

lemma *sorted-list-of-fmap-is-insort-key-fst*:
assumes m $$ k = *None*
shows *sorted-list-of-fmap* ($m(k$ $$:= $v)$) = *insort-key fst* (k, v) (*sorted-list-of-fmap* m)
⟨*proof*⟩

lemma *distinct-fst-inj*:
assumes *distinct* (*map fst ps*)
and *inj f*
shows *distinct* (*map fst* (*map* (λ(k, v). (*f k*, v)) *ps*))
⟨*proof*⟩

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

shows *distinct* (*map fst* (*sorted-list-of-fmap m*))
⟨*proof*⟩

lemma *map-inj-pair-non-membership*:
assumes $k \notin set\ (map\ fst\ ps)$
and *inj f*
shows $f\ k \notin set\ (map\ fst\ (map\ (\lambda(k,\ v).\ (f\ k,\ v))\ ps))$
⟨*proof*⟩

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

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

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

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

lemma *fmap-of-list-sorted-list-of-fmap*:
fixes $m :: ('a::linorder,\ 'b)\ fmap$
and $f :: 'a \Rightarrow 'c::linorder$
assumes *inj f*
and *mono f*
and $m\ \$\$\ k = None$
shows $fmap\text{-}of\text{-}list\ (map\ (\lambda(k,\ v).\ (f\ k,\ v))\ (sorted\text{-}list\text{-}of\text{-}fmap\ (m(k\ \$\$:=\ v)))) =$
$(fmap\text{-}of\text{-}list\ (map\ (\lambda(k,\ v).\ (f\ k,\ v))\ (sorted\text{-}list\text{-}of\text{-}fmap\ m)))(f\ k\ \$\$:=\ v)$
⟨*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* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* (**infixl** ‹$--_f$› *100*) **where**
$m_1 --_f m_2 \equiv$ *fmfilter* ($\lambda x.\ x \notin$ *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* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* (**infixl** ‹Δ_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 set* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* (**infixl** ‹$\lhd$› *150*) **where**
$s \lhd m \equiv$ *fmfilter* ($\lambda x.\ x \in s$) m

Domain exclusion

abbreviation *dom-exc* :: '*a set* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* (**infixl** ‹$\lhd/$› *150*) **where**
$s \lhd/ m \equiv$ *fmfilter* ($\lambda x.\ x \notin s$) m

Intersection plus

abbreviation *intersection-plus* :: ('*a*, '*b*::*monoid-add*) *fmap* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap*
(**infixl** ‹$\cap_+$› *100*)
where
$m_1 \cap_+ m_2 \equiv$ *fmmap-keys* ($\lambda k\ v.\ v + m_1$ $\$\$!$ k) (*fmdom'* $m_1 \lhd m_2$)

Union override right

abbreviation *union-override-right* :: ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap*
(**infixl** ‹$\cup_\rightarrow$› *100*)
where
$m_1 \cup_\rightarrow m_2 \equiv$ (*fmdom'* $m_2 \lhd/ m_1$) $++_f m_2$

Union override left

abbreviation *union-override-left* :: ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap*
(**infixl** ‹$\cup_\leftarrow$› *100*)
where

$m_1 \cup_{\leftarrow} m_2 \equiv m_1 +\!\!+_f (fmdom' m_1 \lhd/ m_2)$

Union override plus

abbreviation *union-override-plus* :: ('*a*, '*b*::*monoid-add*) *fmap* ⇒ ('*a*, '*b*) *fmap* ⇒ ('*a*, '*b*) *fmap*
(**infixl** ‹$\cup_+$› *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 $m \$\$ k = Some\ v$
shows $\{k\} \lhd m = \{k \$\$:= v\}$
⟨*proof*⟩

lemma *dom-res-union-distr*:
shows $(A \cup B) \lhd m = A \lhd m +\!\!+_f B \lhd m$
⟨*proof*⟩

lemma *dom-exc-add-distr*:
shows $s \lhd/ (m_1 +\!\!+_f m_2) = (s \lhd/ m_1) +\!\!+_f (s \lhd/ m_2)$
⟨*proof*⟩

lemma *fmap-partition*:
shows $m = s \lhd/ m +\!\!+_f s \lhd m$
⟨*proof*⟩

lemma *dom-res-addition-in*:
assumes $m_1 \$\$ k = None$
and $m_2 \$\$ k = Some\ v'$
shows $fmdom' (m_1(k \$\$:= v)) \lhd m_2 = fmdom' m_1 \lhd m_2 +\!\!+_f \{k \$\$:= v'\}$
⟨*proof*⟩

lemma *dom-res-addition-not-in*:
assumes $m_2 \$\$ k = None$
shows $fmdom' (m_1(k \$\$:= v)) \lhd m_2 = fmdom' m_1 \lhd m_2$
⟨*proof*⟩

lemma *inter-plus-addition-in*:
assumes $m_1 \$\$ k = None$
and $m_2 \$\$ k = Some\ v'$
shows $m_1(k \$\$:= v) \cap_+ m_2 = (m_1 \cap_+ m_2) +\!\!+_f \{k \$\$:= v' + v\}$
⟨*proof*⟩

lemma *inter-plus-addition-notin*:
assumes $m_1 \$\$ k = None$
and $m_2 \$\$ k = None$
shows $m_1(k \$\$:= v) \cap_+ m_2 = (m_1 \cap_+ m_2)$
⟨*proof*⟩

lemma *union-plus-addition-notin*:
  assumes m_1 $\$\$$ $k = None$
  and m_2 $\$\$$ $k = None$
  shows $m_1(k \mathbin{\$\$:=} v) \cup_+ m_2 = (m_1 \cup_+ m_2) +\!\!+_f \{k \mathbin{\$\$:=} v\}$
⟨*proof*⟩

end