Minimal, Maximal, Least, and Greatest Elements w.r.t. Restricted Ordering

Martin Desharnais

March 25, 2025

Abstract

This entry provides small, reusable, theories that specify the concepts of minimal, maximal, least, and greatest elements in sets, final sets, and final multisets. The concepts are uniformly specified as predicates parametrized by a binary relation. The binary relation is only required to be an ordering on the elements of the concrete collection considered. This is useful when working with a partial ordering, but some assumption or invariant proves that the ordering is total on all elements of the considered set.

Contents

theory *Relation-Reachability*
imports *Main*
begin

1 Definitions

When a binary relation hold for two values, i.e., $R\ x\ y$, we say that x reaches y and, conversely, that y is reachable by x.

definition *non-reachable-wrt* **where**
$non\text{-}reachable\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X - \{x\}.\ \neg\ (R\ y\ x))$

definition *non-reaching-wrt* **where**
$non\text{-}reaching\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X - \{x\}.\ \neg\ (R\ x\ y))$

definition *reaching-all-wrt* **where**
$reaching\text{-}all\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X - \{x\}.\ R\ x\ y)$

definition *reachable-by-all-wrt* **where**
$reachable\text{-}by\text{-}all\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X - \{x\}.\ R\ y\ x)$

2 Conversions

lemma *non-reachable-wrt-iff*:
$non\text{-}reachable\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X.\ y \neq x \longrightarrow \neg\ R\ y\ x)$
unfolding *non-reachable-wrt-def* **by** *blast*

lemma *non-reaching-wrt-iff*:
$non\text{-}reaching\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X.\ y \neq x \longrightarrow \neg\ R\ x\ y)$
unfolding *non-reaching-wrt-def* **by** *blast*

lemma *reaching-all-wrt-iff*:
$reaching\text{-}all\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X.\ y \neq x \longrightarrow R\ x\ y)$
unfolding *reaching-all-wrt-def* **by** *blast*

lemma *reachable-by-all-wrt-iff*:
$reachable\text{-}by\text{-}all\text{-}wrt\ R\ X\ x \longleftrightarrow x \in X \wedge (\forall\, y \in X.\ y \neq x \longrightarrow R\ y\ x)$
unfolding *reachable-by-all-wrt-def* **by** *blast*

lemma *non-reachable-wrt-filter-iff*:
$non\text{-}reachable\text{-}wrt\ R\ \{y \in X.\ P\ y\}\ x \longleftrightarrow x \in X \wedge P\ x \wedge (\forall\, y \in X - \{x\}.\ P\ y \longrightarrow \neg\ R\ y\ x)$
by (*auto simp*: *non-reachable-wrt-def*)

lemma *non-reachable-wrt-conversep*[*simp*]:
$non\text{-}reachable\text{-}wrt\ R^{-1-1} = non\text{-}reaching\text{-}wrt\ R$
unfolding *non-reaching-wrt-def non-reachable-wrt-def* **by** *simp*

lemma *non-reaching-wrt-conversep*[*simp*]:
$non\text{-}reaching\text{-}wrt\ R^{-1-1} = non\text{-}reachable\text{-}wrt\ R$
unfolding *non-reaching-wrt-def non-reachable-wrt-def* **by** *simp*

lemma *reaching-all-wrt-conversep*[*simp*]:
$reaching\text{-}all\text{-}wrt\ R^{-1-1} = reachable\text{-}by\text{-}all\text{-}wrt\ R$
unfolding *reaching-all-wrt-def reachable-by-all-wrt-def* **by** *simp*

lemma *reachable-by-all-wrt-conversep*[*simp*]:
$reachable\text{-}by\text{-}all\text{-}wrt\ R^{-1-1} = reaching\text{-}all\text{-}wrt\ R$

unfolding *reaching-all-wrt-def reachable-by-all-wrt-def* **by** *simp*

lemma *non-reachable-wrt-eq-reaching-all-wrt*:
assumes *asym*: *asymp-on X R* **and** *tot*: *totalp-on X R*
shows *non-reachable-wrt R X* = *reaching-all-wrt R X*
proof (*intro ext iffI*)
fix *x*
from *tot* **show** *non-reachable-wrt R X x* $\Longrightarrow$ *reaching-all-wrt R X x*
unfolding *non-reachable-wrt-def reaching-all-wrt-def*
by (*metis Diff-iff insertCI totalp-onD*)
next
fix *x*
from *asym* **show** *reaching-all-wrt R X x* $\Longrightarrow$ *non-reachable-wrt R X x*
unfolding *reaching-all-wrt-def non-reachable-wrt-def*
by (*metis Diff-iff asymp-onD*)
qed

lemma *non-reaching-wrt-eq-reachable-by-all-wrt*:
assumes *asym*: *asymp-on X R* **and** *tot*: *totalp-on X R*
shows *non-reaching-wrt R X* = *reachable-by-all-wrt R X*
proof (*intro ext iffI*)
fix *x*
from *tot* **show** *non-reaching-wrt R X x* $\Longrightarrow$ *reachable-by-all-wrt R X x*
unfolding *non-reaching-wrt-def reachable-by-all-wrt-def*
by (*metis Diff-iff insertCI totalp-onD*)
next
fix *x*
from *asym* **show** *reachable-by-all-wrt R X x* $\Longrightarrow$ *non-reaching-wrt R X x*
unfolding *reachable-by-all-wrt-def non-reaching-wrt-def*
by (*metis Diff-iff asymp-onD*)
qed

lemma *non-reachable-wrt-reflclp*[*simp*]:
non-reachable-wrt $R^{==}$ = *non-reachable-wrt R*
by (*intro ext iffI*) (*simp-all add*: *non-reachable-wrt-iff*)

lemma *non-reaching-wrt-reflclp*[*simp*]:
non-reaching-wrt $R^{==}$ = *non-reaching-wrt R*
by (*intro ext iffI*) (*simp-all add*: *non-reaching-wrt-iff*)

lemma *reaching-all-wrt-reflclp*[*simp*]:
reaching-all-wrt $R^{==}$ = *reaching-all-wrt R*
by (*intro ext iffI*) (*simp-all add*: *reaching-all-wrt-iff*)

lemma *reachable-by-all-wrt-reflclp*[*simp*]:
reachable-by-all-wrt $R^{==}$ = *reachable-by-all-wrt R*
by (*intro ext iffI*) (*simp-all add*: *reachable-by-all-wrt-iff*)

3 Existence

lemma *ex-non-reachable-wrt*:
transp-on A $R \Longrightarrow$ *asymp-on* A $R \Longrightarrow$ *finite* $A \Longrightarrow A \neq \{\} \Longrightarrow \exists\, m.$ *non-reachable-wrt* R A m
using *Finite-Set.bex-min-element*
by (*metis non-reachable-wrt-iff*)

lemma *ex-non-reaching-wrt*:
transp-on A $R \Longrightarrow$ *asymp-on* A $R \Longrightarrow$ *finite* $A \Longrightarrow A \neq \{\} \Longrightarrow \exists\, m.$ *non-reaching-wrt* R A m
using *Finite-Set.bex-max-element*
by (*metis non-reaching-wrt-iff*)

lemma *ex-reaching-all-wrt*:
transp-on A $R \Longrightarrow$ *totalp-on* A $R \Longrightarrow$ *finite* $A \Longrightarrow A \neq \{\} \Longrightarrow \exists\, g.$ *reaching-all-wrt* R A g
using *Finite-Set.bex-least-element*[*of* A R]
by (*metis reaching-all-wrt-iff*)

lemma *ex-reachable-by-all-wrt*:
transp-on A $R \Longrightarrow$ *totalp-on* A $R \Longrightarrow$ *finite* $A \Longrightarrow A \neq \{\} \Longrightarrow \exists\, g.$ *reachable-by-all-wrt* R A g
using *Finite-Set.bex-greatest-element*[*of* A R]
by (*metis reachable-by-all-wrt-iff*)

lemma *not-ex-greatest-element-doubleton-if*:
assumes $x \neq y$ **and** $\neg$ R x y **and** $\neg$ R y x
shows $\nexists\, g.$ *reachable-by-all-wrt* R $\{x, y\}$ g
proof (*rule notI*)
assume $\exists\, g.$ *reachable-by-all-wrt* R $\{x, y\}$ g
then obtain g **where** *reachable-by-all-wrt* R $\{x, y\}$ g **..**
then show *False*
unfolding *reachable-by-all-wrt-def*
using *assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *blast*
qed

4 Uniqueness

lemma *Uniq-non-reachable-wrt*:
totalp-on X $R \Longrightarrow \exists_{\leq 1} x.$ *non-reachable-wrt* R X x
by (*rule Uniq-I*) (*metis insert-Diff insert-iff non-reachable-wrt-def totalp-onD*)

lemma *Uniq-non-reaching-wrt*:
totalp-on X $R \Longrightarrow \exists_{\leq 1} x.$ *non-reaching-wrt* R X x
by (*rule Uniq-I*) (*metis insert-Diff insert-iff non-reaching-wrt-def totalp-onD*)

lemma *Uniq-reaching-all-wrt*:
asymp-on X $R \Longrightarrow \exists_{\leq 1} x.$ *reaching-all-wrt* R X x

by (*rule Uniq-I*)
(*metis antisymp-onD antisymp-on-if-asymp-on insertE insert-Diff reaching-all-wrt-def*)

lemma *Uniq-reachable-by-all-wrt*:
asymp-on X R $\Longrightarrow \exists_{\leq 1} x.$ *reachable-by-all-wrt R X x*
by (*rule Uniq-I*)
(*metis antisymp-onD antisymp-on-if-asymp-on insertE insert-Diff reachable-by-all-wrt-def*)

5 Existence of unique element

lemma *ex1-reaching-all-wrt*:
transp-on X R $\Longrightarrow$ *asymp-on X R* $\Longrightarrow$ *totalp-on X R* $\Longrightarrow$ *finite X* $\Longrightarrow X \neq \{\}$
$\Longrightarrow$
$\exists!x.$ *reaching-all-wrt R X x*
using *ex1-iff-ex-Uniq ex-reaching-all-wrt Uniq-reaching-all-wrt* **by** *metis*

lemma *ex1-reachable-by-all-wrt*:
transp-on X R $\Longrightarrow$ *asymp-on X R* $\Longrightarrow$ *totalp-on X R* $\Longrightarrow$ *finite X* $\Longrightarrow X \neq \{\}$
$\Longrightarrow$
$\exists!x.$ *reachable-by-all-wrt R X x*
using *ex1-iff-ex-Uniq ex-reachable-by-all-wrt Uniq-reachable-by-all-wrt* **by** *metis*

6 Transformations

lemma *non-reachable-wrt-insert-wrtI*:
assumes
trans: *transp-on* (*insert y X*) *R* **and**
asym: *asymp-on* (*insert y X*) *R* **and**
R y x **and**
x-non-reachable: *non-reachable-wrt R X x*
shows *non-reachable-wrt R* (*insert y X*) *y*
proof −
from *x-non-reachable* **have** *x-in*: $x \in X$ **and** *x-min'*: $\forall y \in X - \{x\}. \neg R\ y\ x$
by (*simp-all add*: *non-reachable-wrt-iff*)

have $\neg R\ z\ y$ **if** $z \in$ *insert y X* $- \{y\}$ **for** z
proof −
from *that* **have** $z \in X$ **and** $z \neq y$
by *simp-all*

show $\neg R\ z\ y$
proof (*cases* $z = x$)
case *True*
thus *?thesis*
using ‹*R y x*› *asym x-in*
by (*metis asymp-onD insertI1 insertI2*)
next
case *False*

hence $\neg$ *R z x*
using *x-min*′[*rule-format*, *of z*, *simplified*] ‹*z* ∈ *X*› **by** *metis*
then show *?thesis*
using ‹*R y x*› *trans* ‹*z* ∈ *X*› *x-in*
by (*meson insertCI transp-onD*)
qed
qed
thus *?thesis*
by (*simp add*: *non-reachable-wrt-def*)
qed

end
theory *Min-Max-Least-Greatest-Set*
imports
Relation-Reachability
begin

7 Minimal and maximal elements

If the binary relation is a strict partial order, then non-reachability corresponds to minimality and non-reaching correspond to maximality.

definition *is-minimal-in-set-wrt* :: ($'a \Rightarrow 'a \Rightarrow bool$) $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a \Rightarrow bool$ **where**
transp-on X R $\Longrightarrow$ *asymp-on X R* $\Longrightarrow$ *is-minimal-in-set-wrt R X* = *non-reachable-wrt R X*

definition *is-maximal-in-set-wrt* :: ($'a \Rightarrow 'a \Rightarrow bool$) $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a \Rightarrow bool$ **where**
transp-on X R $\Longrightarrow$ *asymp-on X R* $\Longrightarrow$ *is-maximal-in-set-wrt R X* = *non-reaching-wrt R X*

context
fixes *X R*
assumes
trans: *transp-on X R* **and**
asym: *asymp-on X R*
begin

7.1 Conversions

lemma *is-minimal-in-set-wrt-iff*:
is-minimal-in-set-wrt R X x $\longleftrightarrow$ $x \in X \land (\forall y \in X.\ y \neq x \longrightarrow \neg R\ y\ x)$
using *trans asym is-minimal-in-set-wrt-def*[*unfolded non-reachable-wrt-iff*] **by** *metis*

lemma *is-maximal-in-set-wrt-iff*:
is-maximal-in-set-wrt R X x $\longleftrightarrow$ $x \in X \land (\forall y \in X.\ y \neq x \longrightarrow \neg R\ x\ y)$
using *trans asym is-maximal-in-set-wrt-def*[*unfolded non-reaching-wrt-iff*] **by** *metis*

7.2 Existence

lemma *ex-minimal-in-set-wrt*:
  finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists x.$ *is-minimal-in-set-wrt* $R\ X\ x$
  using *trans asym ex-non-reachable-wrt is-minimal-in-set-wrt-def* **by** *metis*

lemma *ex-maximal-in-set-wrt*:
  finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists m.$ *is-maximal-in-set-wrt* $R\ X\ m$
  using *trans asym is-maximal-in-set-wrt-def ex-non-reaching-wrt* **by** *metis*

end

7.3 Miscellaneous

lemma *is-minimal-in-set-wrt-filter-iff*:
  fixes $X\ R$
  assumes *trans*: *transp-on* $\{y \in X.\ P\ y\}\ R$ **and** *asym*: *asymp-on* $\{y \in X.\ P\ y\}$ R
  shows *is-minimal-in-set-wrt* $R\ \{y \in X.\ P\ y\}\ x \longleftrightarrow x \in X \wedge P\ x \wedge (\forall y \in X - \{x\}.\ P\ y \longrightarrow \neg\ R\ y\ x)$
proof −
  have *is-minimal-in-set-wrt* $R\ \{y \in X.\ P\ y\}\ x \longleftrightarrow$ *non-reachable-wrt* $R\ \{y \in X.\ P\ y\}\ x$
    using *trans asym is-minimal-in-set-wrt-def* **by** *metis*
  also have $\ldots \longleftrightarrow x \in X \wedge P\ x \wedge (\forall y \in X - \{x\}.\ P\ y \longrightarrow \neg\ R\ y\ x)$
    unfolding *non-reachable-wrt-filter-iff* **..**
  finally show *?thesis* **.**
qed

lemma *is-minimal-in-set-wrt-insertI*:
  assumes
    trans: *transp-on* (*insert* $y\ X$) R **and**
    asym: *asymp-on* (*insert* $y\ X$) R **and**
    $R\ y\ x$ **and**
    x-min: *is-minimal-in-set-wrt* $R\ X\ x$
  shows *is-minimal-in-set-wrt* R (*insert* $y\ X$) y
  by (*metis assms*(*3*) *asym asymp-on-subset is-minimal-in-set-wrt-def trans non-reachable-wrt-insert-wrtI subset-insertI transp-on-subset x-min*)

8 Least and greatest elements

If the binary relation is a strict total ordering, then an element reaching all others is a least element and an element reachable by all others is a greatest element.

definition *is-least-in-set-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow 'a \Rightarrow bool$ **where**
  transp-on $X\ R \Longrightarrow$ *asymp-on* $X\ R \Longrightarrow$ *totalp-on* $X\ R \Longrightarrow$
    is-least-in-set-wrt $R\ X =$ *reaching-all-wrt* $R\ X$

definition *is-greatest-in-set-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow 'a \Rightarrow bool$ **where**
transp-on X R $\Longrightarrow$ *asymp-on X R* $\Longrightarrow$ *totalp-on X R* $\Longrightarrow$
is-greatest-in-set-wrt R X = *reachable-by-all-wrt R X*

context
fixes *X R*
assumes
trans: *transp-on X R* **and**
asym: *asymp-on X R* **and**
tot: *totalp-on X R*
begin

8.1 Conversions

lemma *is-least-in-set-wrt-iff*:
is-least-in-set-wrt R X x $\longleftrightarrow x \in X \wedge (\forall y \in X.\ y \neq x \longrightarrow R\ x\ y)$
using *trans asym tot is-least-in-set-wrt-def*[*unfolded reaching-all-wrt-iff*] **by** *metis*

lemma *is-greatest-in-set-wrt-iff*:
is-greatest-in-set-wrt R X x $\longleftrightarrow x \in X \wedge (\forall y \in X.\ y \neq x \longrightarrow R\ y\ x)$
using *trans asym tot is-greatest-in-set-wrt-def*[*unfolded reachable-by-all-wrt-iff*]
by *metis*

lemma *is-minimal-in-set-wrt-eq-is-least-in-set-wrt*:
is-minimal-in-set-wrt R X = *is-least-in-set-wrt R X*
using *trans asym tot non-reachable-wrt-eq-reaching-all-wrt*
is-minimal-in-set-wrt-def is-least-in-set-wrt-def
by *metis*

lemma *is-maximal-in-set-wrt-eq-is-greatest-in-set-wrt*:
is-maximal-in-set-wrt R X = *is-greatest-in-set-wrt R X*
using *trans asym tot non-reaching-wrt-eq-reachable-by-all-wrt*
is-maximal-in-set-wrt-def is-greatest-in-set-wrt-def
by *metis*

8.2 Uniqueness

lemma *Uniq-is-least-in-set-wrt*:
$\exists_{\leq 1} x.$ *is-least-in-set-wrt R X x*
using *trans asym tot is-least-in-set-wrt-def Uniq-reaching-all-wrt* **by** *metis*

lemma *Uniq-is-greatest-in-set-wrt*:
$\exists_{\leq 1} x.$ *is-greatest-in-set-wrt R X x*
using *trans asym tot is-greatest-in-set-wrt-def Uniq-reachable-by-all-wrt* **by** *metis*

8.3 Existence

lemma *ex-least-in-set-wrt*:
finite X $\Longrightarrow X \neq \{\} \Longrightarrow \exists x.$ *is-least-in-set-wrt R X x*
using *trans asym tot is-least-in-set-wrt-def ex-reaching-all-wrt* **by** *metis*

lemma *ex-greatest-in-set-wrt*:
finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists\, x.$ *is-greatest-in-set-wrt* $R\ X\ x$
using *trans asym tot is-greatest-in-set-wrt-def ex-reachable-by-all-wrt* **by** *metis*

lemma *ex1-least-in-set-wrt*:
finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists!x.$ *is-least-in-set-wrt* $R\ X\ x$
using *Uniq-is-least-in-set-wrt ex-least-in-set-wrt* **by** (*metis Uniq-def*)

lemma *ex1-greatest-in-set-wrt*:
finite $X \Longrightarrow X \neq \{\} \Longrightarrow \exists!x.$ *is-greatest-in-set-wrt* $R\ X\ x$
using *Uniq-is-greatest-in-set-wrt ex-greatest-in-set-wrt* **by** (*metis Uniq-def*)

end

9 Hide stuff

We restrict the public interface to ease future internal changes.

hide-fact *is-minimal-in-set-wrt-def is-maximal-in-set-wrt-def*
hide-fact *is-least-in-set-wrt-def is-greatest-in-set-wrt-def*

10 Integration in type classes

abbreviation (**in** *order*) *is-minimal-in-set* **where**
is-minimal-in-set $\equiv$ *is-minimal-in-set-wrt* $(<)$

abbreviation (**in** *order*) *is-maximal-in-set* **where**
is-maximal-in-set $\equiv$ *is-maximal-in-set-wrt* $(<)$

lemmas (**in** *order*) *is-minimal-in-set-iff* =
is-minimal-in-set-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-maximal-in-set-iff* =
is-maximal-in-set-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-minimal-in-set* =
ex-minimal-in-set-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-maximal-in-set* =
ex-maximal-in-set-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-minimal-in-set-filter-iff* =
is-minimal-in-set-wrt-filter-iff[*OF transp-on-less asymp-on-less*]

abbreviation (**in** *linorder*) *is-least-in-set* **where**
is-least-in-set $\equiv$ *is-least-in-set-wrt* $(<)$

abbreviation (**in** *linorder*) *is-greatest-in-set* **where**
is-greatest-in-set ≡ *is-greatest-in-set-wrt* (<)

lemmas (**in** *linorder*) *is-least-in-set-iff* =
is-least-in-set-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-greatest-in-set-iff* =
is-greatest-in-set-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-minimal-in-set-eq-is-least-in-set* =
is-minimal-in-set-wrt-eq-is-least-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-maximal-in-set-eq-is-greatest-in-set* =
is-maximal-in-set-wrt-eq-is-greatest-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-least-in-set*[*intro*] =
Uniq-is-least-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-greatest-in-set*[*intro*] =
Uniq-is-greatest-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex-least-in-set* =
ex-least-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex-greatest-in-set* =
ex-greatest-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex1-least-in-set* =
ex1-least-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex1-greatest-in-set* =
ex1-greatest-in-set-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

end
theory *Min-Max-Least-Greatest-FSet*
imports
Min-Max-Least-Greatest-Set
HOL−Library.FSet
begin

11 Minimal and maximal elements

definition *is-minimal-in-fset-wrt* :: (*'a* ⇒ *'a* ⇒ *bool*) ⇒ *'a fset* ⇒ *'a* ⇒ *bool* **where**
transp-on (*fset X*) *R* ⟹ *asymp-on* (*fset X*) *R* ⟹
is-minimal-in-fset-wrt R X = *is-minimal-in-set-wrt R* (*fset X*)

definition *is-maximal-in-fset-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ fset \Rightarrow 'a \Rightarrow bool$
where
  $transp\text{-}on\ (fset\ X)\ R \Longrightarrow asymp\text{-}on\ (fset\ X)\ R \Longrightarrow$
    $is\text{-}maximal\text{-}in\text{-}fset\text{-}wrt\ R\ X = is\text{-}maximal\text{-}in\text{-}set\text{-}wrt\ R\ (fset\ X)$

context
  fixes $X\ R$
  assumes
    trans: $transp\text{-}on\ (fset\ X)\ R$ **and**
    asym: $asymp\text{-}on\ (fset\ X)\ R$
begin

11.1 Conversions

lemma *is-minimal-in-fset-wrt-iff*:
  $is\text{-}minimal\text{-}in\text{-}fset\text{-}wrt\ R\ X\ x \longleftrightarrow x\ |\in|\ X \wedge fBall\ X\ (\lambda y.\ y \neq x \longrightarrow \neg\ R\ y\ x)$
  using *is-minimal-in-set-wrt-iff*[*OF trans asym*]
  using *is-minimal-in-fset-wrt-def*[*OF trans asym*]
  by *simp*

lemma *is-maximal-in-fset-wrt-iff*:
  $is\text{-}maximal\text{-}in\text{-}fset\text{-}wrt\ R\ X\ x \longleftrightarrow x\ |\in|\ X \wedge fBall\ X\ (\lambda y.\ y \neq x \longrightarrow \neg\ R\ x\ y)$
  using *is-maximal-in-set-wrt-iff*[*OF trans asym*]
  using *is-maximal-in-fset-wrt-def*[*OF trans asym*]
  by *simp*

11.2 Existence

lemma *ex-minimal-in-fset-wrt*:
  $X \neq \{||\} \Longrightarrow \exists\, m.\ is\text{-}minimal\text{-}in\text{-}fset\text{-}wrt\ R\ X\ m$
  using *trans asym ex-minimal-in-set-wrt*[*of fset X R*] *is-minimal-in-fset-wrt-def*
  by (*metis all-not-fin-conv empty-iff finite-fset*)

lemma *ex-maximal-in-fset-wrt*:
  $X \neq \{||\} \Longrightarrow \exists\, m.\ is\text{-}maximal\text{-}in\text{-}fset\text{-}wrt\ R\ X\ m$
  using *trans asym ex-maximal-in-set-wrt*[*of fset X R*] *is-maximal-in-fset-wrt-def*
  by (*metis all-not-fin-conv empty-iff finite-fset*)

end

11.3 Non-existence

lemma *not-is-minimal-in-fset-wrt-fempty*[*simp*]: $\bigwedge R\ x.\ \neg\ is\text{-}minimal\text{-}in\text{-}fset\text{-}wrt\ R\ \{||\}\ x$
  using *is-minimal-in-fset-wrt-iff*[*of* $\{||\}$]
  by (*simp add*: *transp-on-def asymp-on-def*)

lemma *not-is-maximal-in-fset-wrt-fempty*[*simp*]: $\bigwedge R\ x.\ \neg\ is\text{-}maximal\text{-}in\text{-}fset\text{-}wrt\ R\ \{||\}\ x$
  using *is-maximal-in-fset-wrt-iff*[*of* $\{||\}$]

by (*simp add*: *transp-on-def asymp-on-def*)

11.4 Miscellaneous

lemma *is-minimal-in-fset-wrt-ffilter-iff*:
assumes
tran: *transp-on* (*fset* (*ffilter* P X)) R **and**
asym: *asymp-on* (*fset* (*ffilter* P X)) R
shows *is-minimal-in-fset-wrt* R (*ffilter* P X) $x \longleftrightarrow$
(x |∈| $X \wedge P\ x \wedge$ *fBall* ($X - \{|x|\}$) ($\lambda y.\ P\ y \longrightarrow \neg\ R\ y\ x$))
proof −
have *is-minimal-in-fset-wrt* R (*ffilter* P X) $x \longleftrightarrow$ *is-minimal-in-set-wrt* R ($\{y \in$ *fset* $X.\ P\ y\}$) x
using *is-minimal-in-fset-wrt-iff*[*OF tran asym*]
using *is-minimal-in-set-wrt-iff*[*OF tran asym*]
by (*simp only*: *ffilter.rep-eq Set.filter-def*)
also have ... $\longleftrightarrow x$ |∈| $X \wedge P\ x \wedge (\forall y \in$*fset* $X - \{x\}.\ P\ y \longrightarrow \neg\ R\ y\ x)$
proof (*rule is-minimal-in-set-wrt-filter-iff*)
show *transp-on* $\{y.\ y$ |∈| $X \wedge P\ y\}$ R
using *tran ffilter.rep-eq Set.filter-def* **by** *metis*
next
show *asymp-on* $\{y.\ y$ |∈| $X \wedge P\ y\}$ R
using *asym ffilter.rep-eq Set.filter-def* **by** *metis*
qed
finally show *?thesis*
by *simp*
qed

lemma *is-minimal-in-fset-wrt-finsertI*:
assumes *trans*: *transp-on* (*fset* (*finsert* y X)) R **and** *asym*: *asymp-on* (*fset* (*finsert* y X)) R
shows $R\ y\ x \Longrightarrow$ *is-minimal-in-fset-wrt* R X $x \Longrightarrow$ *is-minimal-in-fset-wrt* R (*finsert* y X) y
using *trans asym is-minimal-in-set-wrt-insertI*[*of - fset -*, *folded fset-simps*]
by (*smt* (*verit*) *asymp-on-def finsertCI finsertE is-minimal-in-fset-wrt-iff transp-on-def*)

12 Least and greatest elements

definition *is-least-in-fset-wrt* :: ($'a \Rightarrow 'a \Rightarrow$ *bool*) $\Rightarrow$ $'a$ *fset* $\Rightarrow$ $'a \Rightarrow$ *bool* **where**
transp-on (*fset* X) $R \Longrightarrow$ *asymp-on* (*fset* X) $R \Longrightarrow$ *totalp-on* (*fset* X) $R \Longrightarrow$
is-least-in-fset-wrt R X = *is-least-in-set-wrt* R (*fset* X)

definition *is-greatest-in-fset-wrt* :: ($'a \Rightarrow 'a \Rightarrow$ *bool*) $\Rightarrow$ $'a$ *fset* $\Rightarrow$ $'a \Rightarrow$ *bool* **where**
transp-on (*fset* X) $R \Longrightarrow$ *asymp-on* (*fset* X) $R \Longrightarrow$ *totalp-on* (*fset* X) $R \Longrightarrow$
is-greatest-in-fset-wrt R X = *is-greatest-in-set-wrt* R (*fset* X)

context
fixes X R
assumes

trans: *transp-on* (*fset* X) R **and**
asym: *asymp-on* (*fset* X) R **and**
tot: *totalp-on* (*fset* X) R
begin

12.1 Conversions

lemma *is-least-in-fset-wrt-iff*:
is-least-in-fset-wrt R X x $\longleftrightarrow$ x $|\in|$ X $\wedge$ *fBall* X ($\lambda y.$ $y \neq x \longrightarrow R\ x\ y$)
using *is-least-in-set-wrt-iff*[*OF trans asym tot*]
using *is-least-in-fset-wrt-def*[*OF trans asym tot*]
by *simp*

lemma *is-greatest-in-fset-wrt-iff*:
is-greatest-in-fset-wrt R X x $\longleftrightarrow$ x $|\in|$ X $\wedge$ *fBall* X ($\lambda y.$ $y \neq x \longrightarrow R\ y\ x$)
using *is-greatest-in-set-wrt-iff*[*OF trans asym tot*]
using *is-greatest-in-fset-wrt-def*[*OF trans asym tot*]
by *simp*

lemma *is-minimal-in-fset-wrt-eq-is-least-in-fset-wrt*:
is-minimal-in-fset-wrt R X = *is-least-in-fset-wrt* R X
using *trans asym tot is-minimal-in-set-wrt-eq-is-least-in-set-wrt*
by (*metis is-least-in-fset-wrt-def is-minimal-in-fset-wrt-def*)

lemma *is-maximal-in-fset-wrt-eq-is-greatest-in-fset-wrt*:
is-maximal-in-fset-wrt R X = *is-greatest-in-fset-wrt* R X
using *trans asym tot is-maximal-in-set-wrt-eq-is-greatest-in-set-wrt*
by (*metis is-greatest-in-fset-wrt-def is-maximal-in-fset-wrt-def*)

12.2 Uniqueness

lemma *Uniq-is-least-in-fset-wrt*[*intro*]:
$\exists_{\leq 1} x.$ *is-least-in-fset-wrt* R X x
using *trans asym tot Uniq-is-least-in-set-wrt*
by (*metis is-least-in-fset-wrt-def*)

lemma *Uniq-is-greatest-in-fset-wrt*[*intro*]:
$\exists_{\leq 1} x.$ *is-greatest-in-fset-wrt* R X x
using *trans asym tot Uniq-is-greatest-in-set-wrt*
by (*metis is-greatest-in-fset-wrt-def*)

12.3 Existence

lemma *ex-least-in-fset-wrt*:
$X \neq \{||\} \Longrightarrow \exists x.$ *is-least-in-fset-wrt* R X x
using *trans asym tot ex-least-in-set-wrt*
by (*metis bot-fset.rep-eq finite-fset fset-cong is-least-in-fset-wrt-def*)

lemma *ex-greatest-in-fset-wrt*:
$X \neq \{||\} \Longrightarrow \exists x.$ *is-greatest-in-fset-wrt* R X x

using *trans asym tot ex-greatest-in-set-wrt*
by (*metis bot-fset.rep-eq finite-fset fset-cong is-greatest-in-fset-wrt-def*)

lemma *ex1-least-in-fset-wrt*:
$X \neq \{||\} \Longrightarrow \exists!x.$ *is-least-in-fset-wrt R X x*
using *Uniq-is-least-in-fset-wrt ex-least-in-fset-wrt*
by (*metis Uniq-def*)

lemma *ex1-greatest-in-fset-wrt*:
$X \neq \{||\} \Longrightarrow \exists!x.$ *is-greatest-in-fset-wrt R X x*
using *Uniq-is-greatest-in-fset-wrt ex-greatest-in-fset-wrt*
by (*metis Uniq-def*)

end

12.4 Nonexistence

lemma *not-is-least-in-fset-wrt-fempty*[*simp*]: $\bigwedge R\ x.\ \neg$ *is-least-in-fset-wrt R* $\{||\}$ *x*
using *is-least-in-fset-wrt-iff* [*of* $\{||\}$]
by (*simp add*: *transp-on-def asymp-on-def*)

lemma *not-is-greatest-in-fset-wrt-fempty*[*simp*]: $\bigwedge R\ x.\ \neg$ *is-greatest-in-fset-wrt R* $\{||\}$ *x*
using *is-greatest-in-fset-wrt-iff* [*of* $\{||\}$]
by (*simp add*: *transp-on-def asymp-on-def*)

12.5 Miscellaneous

lemma *is-least-in-ffilter-wrt-iff*:
assumes
trans: *transp-on* (*fset* (*ffilter P X*)) *R* **and**
asym: *asymp-on* (*fset* (*ffilter P X*)) *R* **and**
tot: *totalp-on* (*fset* (*ffilter P X*)) *R*
shows *is-least-in-fset-wrt R* (*ffilter P X*) $x \longleftrightarrow$
$(x\ |\in|\ X \wedge P\ x \wedge$ *fBall X* $(\lambda y.\ y \neq x \longrightarrow P\ y \longrightarrow R\ x\ y))$
unfolding *is-least-in-fset-wrt-iff* [*OF trans asym tot*] **by** *auto*

lemma *is-least-in-ffilter-wrt-swap-predicate*:
assumes
trans: *transp-on* (*fset X*) *R* **and**
asym: *asymp-on* (*fset X*) *R* **and**
tot: *totalp-on* (*fset X*) *R*
assumes
y-least: *is-least-in-fset-wrt R* (*ffilter P X*) *y* **and**
same-on-prefix: $\bigwedge x.\ x\ |\in|\ X \Longrightarrow R^{==}\ x\ y \Longrightarrow P\ x \longleftrightarrow Q\ x$
shows *is-least-in-fset-wrt R* (*ffilter Q X*) *y*
proof −
have $\bigwedge P.$ *fset* (*ffilter P X*) $\subseteq$ *fset X*
by *simp*

```
  hence
    linorder-wrt-P:
      transp-on (fset (ffilter P X)) R
      asymp-on (fset (ffilter P X)) R
      totalp-on (fset (ffilter P X)) R and
    linorder-wrt-Q:
      transp-on (fset (ffilter Q X)) R
      asymp-on (fset (ffilter Q X)) R
      totalp-on (fset (ffilter Q X)) R
    unfolding atomize-conj
   using trans asym tot by (metis transp-on-subset asymp-on-subset totalp-on-subset)

  have y |∈| X and P y and y-lt: ∀ z |∈| X. z ≠ y ⟶ P z ⟶ R y z
    using y-least unfolding is-least-in-ffilter-wrt-iff[OF linorder-wrt-P] by argo+

  show ?thesis
    unfolding is-least-in-ffilter-wrt-iff[OF linorder-wrt-Q]
  proof (intro conjI ballI impI)

    show y |∈| X
      using ‹y |∈| X› .

    show Q y
      using same-on-prefix[of y] ‹y |∈| X› ‹P y› by simp

    fix z
    assume z |∈| X and z ≠ y and Q z
    then show R y z
      using y-lt[rule-format, of z]
      using same-on-prefix[of z]
      by (metis ‹y |∈| X› sup2I1 tot totalp-onD)
  qed
qed

lemma ex-is-least-in-ffilter-wrt-iff:
  assumes
    trans: transp-on (fset (ffilter P X)) R and
    asym: asymp-on (fset (ffilter P X)) R and
    tot: totalp-on (fset (ffilter P X)) R
  shows (∃ x. is-least-in-fset-wrt R (ffilter P X) x) ⟷ (∃ x |∈| X. P x)
  by (metis asym bot-fset.rep-eq empty-iff ex-least-in-fset-wrt ffmember-filter
      is-least-in-fset-wrt-iff local.trans tot)
```

13 Hide stuff

We restrict the public interface to ease future internal changes.

```
hide-fact is-minimal-in-fset-wrt-def is-maximal-in-fset-wrt-def
hide-fact is-least-in-fset-wrt-def is-greatest-in-fset-wrt-def
```

14 Integration in type classes

abbreviation (**in** *order*) *is-minimal-in-fset* **where**
is-minimal-in-fset ≡ *is-minimal-in-fset-wrt* (<)

abbreviation (**in** *order*) *is-maximal-in-fset* **where**
is-maximal-in-fset ≡ *is-maximal-in-fset-wrt* (<)

lemmas (**in** *order*) *is-minimal-in-fset-iff* =
is-minimal-in-fset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-maximal-in-fset-iff* =
is-maximal-in-fset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-minimal-in-fset* =
ex-minimal-in-fset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-maximal-in-fset* =
ex-maximal-in-fset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-minimal-in-fset-ffilter-iff* =
is-minimal-in-fset-wrt-ffilter-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-minimal-in-fset-finsertI* =
is-minimal-in-fset-wrt-finsertI[*OF transp-on-less asymp-on-less*]

abbreviation (**in** *linorder*) *is-least-in-fset* **where**
is-least-in-fset ≡ *is-least-in-fset-wrt* (<)

abbreviation (**in** *linorder*) *is-greatest-in-fset* **where**
is-greatest-in-fset ≡ *is-greatest-in-fset-wrt* (<)

lemmas (**in** *linorder*) *is-least-in-fset-iff* =
is-least-in-fset-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-greatest-in-fset-iff* =
is-greatest-in-fset-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-minimal-in-fset-eq-is-least-in-fset* =
is-minimal-in-fset-wrt-eq-is-least-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-maximal-in-fset-eq-is-greatest-in-fset* =
is-maximal-in-fset-wrt-eq-is-greatest-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-least-in-fset*[*intro*] =
Uniq-is-least-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-greatest-in-fset*[*intro*] =
Uniq-is-greatest-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex-least-in-fset* =
ex-least-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex-greatest-in-fset* =
ex-greatest-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex1-least-in-fset* =
ex1-least-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex1-greatest-in-fset* =
ex1-greatest-in-fset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-least-in-ffilter-iff* =
is-least-in-ffilter-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *ex-is-least-in-ffilter-iff* =
ex-is-least-in-ffilter-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-least-in-ffilter-swap-predicate* =
is-least-in-ffilter-wrt-swap-predicate[*OF transp-on-less asymp-on-less totalp-on-less*]

end
theory *Min-Max-Least-Greatest-Multiset*
imports
Relation-Reachability
Min-Max-Least-Greatest-Set
HOL−Library.Multiset
HOL−Library.Multiset-Order
begin

15 Minimal and maximal elements

definition *is-minimal-in-mset-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ multiset \Rightarrow 'a \Rightarrow bool$
where
transp-on (*set-mset X*) *R* $\Longrightarrow$ *asymp-on* (*set-mset X*) *R* $\Longrightarrow$
is-minimal-in-mset-wrt R X = *is-minimal-in-set-wrt R* (*set-mset X*)

definition *is-maximal-in-mset-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ multiset \Rightarrow 'a \Rightarrow bool$
where
transp-on (*set-mset X*) *R* $\Longrightarrow$ *asymp-on* (*set-mset X*) *R* $\Longrightarrow$
is-maximal-in-mset-wrt R X = *is-maximal-in-set-wrt R* (*set-mset X*)

definition *is-strictly-minimal-in-mset-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ multiset \Rightarrow 'a \Rightarrow bool$ **where**
transp-on (*set-mset X*) *R* $\Longrightarrow$ *asymp-on* (*set-mset X*) *R* $\Longrightarrow$

is-strictly-minimal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#\ x\ \#\}.$
$\neg\ (R^{==}\ y\ x))$

definition *is-strictly-maximal-in-mset-wrt* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ multiset \Rightarrow 'a$
$\Rightarrow bool$ **where**
transp-on (*set-mset* X) $R \Longrightarrow$ *asymp-on* (*set-mset* X) $R \Longrightarrow$
is-strictly-maximal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#\ x\ \#\}.$
$\neg\ (R^{==}\ x\ y))$

context
fixes $X\ R$
assumes
trans: *transp-on* (*set-mset* X) R **and**
asym: *asymp-on* (*set-mset* X) R
begin

15.1 Conversions

lemma *is-minimal-in-mset-wrt-iff*:
is-minimal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X.\ y \neq x \longrightarrow \neg\ R\ y\ x)$
using *is-minimal-in-set-wrt-iff*[*OF trans asym*]
using *is-minimal-in-mset-wrt-def*[*OF trans asym*]
by *simp*

lemma *is-minimal-in-mset-wrt* $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X.\ \neg\ R\ y\ x)$
unfolding *is-minimal-in-mset-wrt-iff*
proof (*rule refl-conj-eq*, *rule ball-cong*)
show *set-mset* X = *set-mset* X **..**
next
show $\bigwedge y.\ y \in\#\ X \Longrightarrow (y \neq x \longrightarrow \neg\ R\ y\ x) = (\neg\ R\ y\ x)$
using *asym*[*THEN asymp-onD*] **by** *metis*
qed

lemma *is-maximal-in-mset-wrt-iff*:
is-maximal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X.\ y \neq x \longrightarrow \neg\ R\ x\ y)$
using *is-maximal-in-set-wrt-iff*[*OF trans asym*]
using *is-maximal-in-mset-wrt-def*[*OF trans asym*]
by *simp*

lemma *is-maximal-in-mset-wrt* $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X.\ \neg\ R\ x\ y)$
unfolding *is-maximal-in-mset-wrt-iff*
proof (*rule refl-conj-eq*, *rule ball-cong*)
show *set-mset* X = *set-mset* X **..**
next
show $\bigwedge y.\ y \in\#\ X \Longrightarrow (y \neq x \longrightarrow \neg\ R\ x\ y) = (\neg\ R\ x\ y)$
using *asym*[*THEN asymp-onD*] **by** *metis*
qed

lemma *is-strictly-minimal-in-mset-wrt-iff*:

is-strictly-minimal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\# X \wedge (\forall y \in\# X - \{\# x \#\}. \neg R^{==}\ y\ x)$
unfolding *is-strictly-minimal-in-mset-wrt-def*[*OF trans asym*]
by(*rule refl*)

lemma *is-strictly-maximal-in-mset-wrt-iff*:
is-strictly-maximal-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\# X \wedge (\forall y \in\# X - \{\# x \#\}. \neg R^{==}\ x\ y)$
unfolding *is-strictly-maximal-in-mset-wrt-def*[*OF trans asym*]
by(*rule refl*)

lemma *is-minimal-in-mset-wrt-if-is-strictly-minimal-in-mset-wrt*:
is-strictly-minimal-in-mset-wrt $R\ X\ x \Longrightarrow$ *is-minimal-in-mset-wrt* $R\ X\ x$
unfolding *is-minimal-in-mset-wrt-iff is-strictly-minimal-in-mset-wrt-iff*
using *multi-member-split* **by** *fastforce*

lemma *is-maximal-in-mset-wrt-if-is-strictly-maximal-in-mset-wrt*:
is-strictly-maximal-in-mset-wrt $R\ X\ x \Longrightarrow$ *is-maximal-in-mset-wrt* $R\ X\ x$
unfolding *is-maximal-in-mset-wrt-iff is-strictly-maximal-in-mset-wrt-iff*
using *multi-member-split* **by** *fastforce*

15.2 Existence

lemma *ex-minimal-in-mset-wrt*:
$X \neq \{\#\} \Longrightarrow \exists m.$ *is-minimal-in-mset-wrt* $R\ X\ m$
using *trans asym ex-minimal-in-set-wrt*[*of set-mset X R*] *is-minimal-in-mset-wrt-def*
by (*metis finite-set-mset set-mset-eq-empty-iff*)

lemma *ex-maximal-in-mset-wrt*:
$X \neq \{\#\} \Longrightarrow \exists m.$ *is-maximal-in-mset-wrt* $R\ X\ m$
using *trans asym ex-maximal-in-set-wrt*[*of set-mset X R*] *is-maximal-in-mset-wrt-def*
by (*metis finite-set-mset set-mset-eq-empty-iff*)

15.3 Miscellaneous

lemma *explode-maximal-in-mset-wrt*:
assumes *max*: *is-maximal-in-mset-wrt* $R\ X\ x$
obtains $n :: nat$ **where** *replicate-mset* $(Suc\ n)\ x + \{\#y \in\# X.\ y \neq x\#\} = X$
using *max*[*unfolded is-maximal-in-mset-wrt-iff*]
by (*metis filter-eq-replicate-mset in-countE multiset-partition*)

lemma *explode-strictly-maximal-in-mset-wrt*:
assumes *max*: *is-strictly-maximal-in-mset-wrt* $R\ X\ x$
shows *add-mset* $x\ \{\#y \in\# X.\ y \neq x\#\} = X$
proof −
have $x \in\# X$ **and** $\forall y \in\# X - \{\#x\#\}.\ x \neq y$
using *max* **unfolding** *is-strictly-maximal-in-mset-wrt-iff* **by** *simp-all*

have *add-mset* $x\ (X - \{\#x\#\}) = X$
using $\langle x \in\# X\rangle$ **by** (*metis insert-DiffM*)

moreover have $\{\#y \in\# X.\ y \neq x\#\} = X - \{\#x\#\}$
using ‹$\forall y \in\# X - \{\#x\#\}.\ x \neq y$›
by (*smt* (*verit*, *best*) ‹$x \in\# X$› *add-diff-cancel-left′ diff-subset-eq-self filter-mset-eq-conv insert-DiffM2 set-mset-add-mset-insert set-mset-empty singletonD*)

ultimately show *?thesis*
by (*simp only*:)
qed

end

lemma *is-minimal-in-filter-mset-wrt-iff*:
assumes
tran: *transp-on* (*set-mset* (*filter-mset P X*)) *R* **and**
asym: *asymp-on* (*set-mset* (*filter-mset P X*)) *R*
shows *is-minimal-in-mset-wrt R* (*filter-mset P X*) $x \longleftrightarrow$
$(x \in\# X \wedge P\ x \wedge (\forall y \in\# X - \{\#x\#\}.\ P\ y \longrightarrow \neg\ R\ y\ x))$
proof −
have *is-minimal-in-mset-wrt R* (*filter-mset P X*) $x \longleftrightarrow$
is-minimal-in-set-wrt R $(\{y \in$ *set-mset* $X.\ P\ y\})\ x$
using *is-minimal-in-mset-wrt-iff*[*OF tran asym*]
using *is-minimal-in-set-wrt-iff*[*OF tran asym*]
by *auto*
also have $\ldots \longleftrightarrow x \in\# X \wedge P\ x \wedge (\forall y \in$ *set-mset* $X - \{x\}.\ P\ y \longrightarrow \neg\ R\ y\ x)$
proof (*rule is-minimal-in-set-wrt-filter-iff*)
show *transp-on* $\{y.\ y \in\# X \wedge P\ y\}$ *R*
using *tran* **by** *simp*
next
show *asymp-on* $\{y.\ y \in\# X \wedge P\ y\}$ *R*
using *asym* **by** *simp*
qed
finally show *?thesis*
by (*metis* (*no-types*, *lifting*) *DiffD1 asym asymp-onD at-most-one-mset-mset-diff*
insertE
insert-Diff is-minimal-in-mset-wrt-iff more-than-one-mset-mset-diff tran)
qed

lemma *multp-if-maximal-of-lhs-is-less*:
assumes
trans: *transp R* **and**
asym: *asymp-on* (*set-mset M1*) *R* **and**
tot: *totalp-on* (*set-mset M1* $\cup$ *set-mset M2*) *R* **and**
$x1 \in\# M1$ **and** $x2 \in\# M2$ **and**
is-maximal-in-mset-wrt R M1 x1 **and** *R x1 x2*
shows *multp R M1 M2*
proof (*rule one-step-implies-multp*[*of* - - - $\{\#\}$, *simplified*])
show $M2 \neq \{\#\}$
using ‹$x2 \in\# M2$› **by** *auto*

next
  show $\forall\, k{\in}\#M1.\ \exists\, j{\in}\#M2.\ R\ k\ j$
    using *assms*
    using *is-maximal-in-mset-wrt-iff*[*OF transp-on-subset*[*OF trans subset-UNIV*] *asym*]
    by (*metis Un-iff totalp-onD transpE*)
qed

15.4 Nonuniqueness

lemma
  fixes x :: $'a$ **and** y :: $'a$
  assumes $x \neq y$
  shows
    not-Uniq-is-minimal-in-mset-if-two-distinct-elements:
      $\neg$ ($\forall$ (R :: $'a \Rightarrow\ 'a \Rightarrow$ *bool*) (X :: $'a$ *multiset*).
        transp-on (*set-mset* X) $R \longrightarrow$ *asymp-on* (*set-mset* X) $R \longrightarrow$
        ($\exists_{\leq 1}x$. *is-minimal-in-mset-wrt* R X x)) **and**
    not-Uniq-is-maximal-in-mset-wrt-if-two-distinct-elements:
      $\neg$ ($\forall$ (R :: $'a \Rightarrow\ 'a \Rightarrow$ *bool*) (X :: $'a$ *multiset*).
        transp-on (*set-mset* X) $R \longrightarrow$ *asymp-on* (*set-mset* X) $R \longrightarrow$
        ($\exists_{\leq 1}x$. *is-maximal-in-mset-wrt* R X x)) **and**
    not-Uniq-is-strictly-minimal-in-mset-if-two-distinct-elements:
      $\neg$ ($\forall$ (R :: $'a \Rightarrow\ 'a \Rightarrow$ *bool*) (X :: $'a$ *multiset*).
        transp-on (*set-mset* X) $R \longrightarrow$ *asymp-on* (*set-mset* X) $R \longrightarrow$
        ($\exists_{\leq 1}x$. *is-strictly-minimal-in-mset-wrt* R X x)) **and**
    not-Uniq-is-strictly-maximal-in-mset-wrt-if-two-distinct-elements:
      $\neg$ ($\forall$ (R :: $'a \Rightarrow\ 'a \Rightarrow$ *bool*) (X :: $'a$ *multiset*).
        transp-on (*set-mset* X) $R \longrightarrow$ *asymp-on* (*set-mset* X) $R \longrightarrow$
        ($\exists_{\leq 1}x$. *is-strictly-maximal-in-mset-wrt* R X x))
proof –
  let *?R* = λ- -. *False*
  let *?X* = {#x, y#}

  have *trans*: *transp-on* (*set-mset ?X*) *?R* **and** *asym*: *asymp-on* (*set-mset ?X*) *?R*
    by (*simp-all add*: *transp-onI asymp-onI*)

  have *is-minimal-in-mset-wrt ?R ?X x* **and** *is-minimal-in-mset-wrt ?R ?X y*
    using *is-minimal-in-mset-wrt-iff*[*OF trans asym*] **by** *simp-all*

  thus $\neg$ ($\forall$ (R :: $'a \Rightarrow\ 'a \Rightarrow$ *bool*) (X :: $'a$ *multiset*).
    transp-on (*set-mset* X) $R \longrightarrow$ *asymp-on* (*set-mset* X) $R \longrightarrow$
    ($\exists_{\leq 1}x$. *is-minimal-in-mset-wrt* R X x))
    using ‹$x \neq y$› *trans asym*
    by (*metis Uniq-D*)

  have *is-maximal-in-mset-wrt ?R ?X x* **and** *is-maximal-in-mset-wrt ?R ?X y*
    using *is-maximal-in-mset-wrt-iff*[*OF trans asym*] **by** *simp-all*

thus ¬ (∀ (*R* :: ′*a* ⇒ ′*a* ⇒ *bool*) (*X* :: ′*a multiset*).
transp-on (*set-mset X*) *R* ⟶ *asymp-on* (*set-mset X*) *R* ⟶
($\exists_{\leq 1}x$. *is-maximal-in-mset-wrt R X x*))
using ‹*x* ≠ *y*› *trans asym*
by (*metis Uniq-D*)

have *is-strictly-minimal-in-mset-wrt ?R ?X x* **and** *is-strictly-minimal-in-mset-wrt ?R ?X y*
using ‹*x* ≠ *y*› *is-strictly-minimal-in-mset-wrt-iff*[*OF trans asym*] **by** *simp-all*

thus ¬ (∀ (*R* :: ′*a* ⇒ ′*a* ⇒ *bool*) (*X* :: ′*a multiset*).
transp-on (*set-mset X*) *R* ⟶ *asymp-on* (*set-mset X*) *R* ⟶
($\exists_{\leq 1}x$. *is-strictly-minimal-in-mset-wrt R X x*))
using ‹*x* ≠ *y*› *trans asym*
by (*metis Uniq-D*)

have *is-strictly-maximal-in-mset-wrt ?R ?X x* **and** *is-strictly-maximal-in-mset-wrt ?R ?X y*
using ‹*x* ≠ *y*› *is-strictly-maximal-in-mset-wrt-iff*[*OF trans asym*] **by** *simp-all*

thus ¬ (∀ (*R* :: ′*a* ⇒ ′*a* ⇒ *bool*) (*X* :: ′*a multiset*).
transp-on (*set-mset X*) *R* ⟶ *asymp-on* (*set-mset X*) *R* ⟶
($\exists_{\leq 1}x$. *is-strictly-maximal-in-mset-wrt R X x*))
using ‹*x* ≠ *y*› *trans asym*
by (*metis Uniq-D*)
qed

16 Least and greatest elements

definition *is-least-in-mset-wrt* :: (′*a* ⇒ ′*a* ⇒ *bool*) ⇒ ′*a multiset* ⇒ ′*a* ⇒ *bool*
where
transp-on (*set-mset X*) *R* ⟹ *asymp-on* (*set-mset X*) *R* ⟹ *totalp-on* (*set-mset X*) *R* ⟹
is-least-in-mset-wrt R X x ⟷ *x* ∈# *X* ∧ (∀ *y* ∈# *X* − {#*x*#}. *R x y*)

definition *is-greatest-in-mset-wrt* :: (′*a* ⇒ ′*a* ⇒ *bool*) ⇒ ′*a multiset* ⇒ ′*a* ⇒ *bool*
where
transp-on (*set-mset X*) *R* ⟹ *asymp-on* (*set-mset X*) *R* ⟹ *totalp-on* (*set-mset X*) *R* ⟹
is-greatest-in-mset-wrt R X x ⟷ *x* ∈# *X* ∧ (∀ *y* ∈# *X* − {#*x*#}. *R y x*)

context
fixes *X R*
assumes
trans: *transp-on* (*set-mset X*) *R* **and**
asym: *asymp-on* (*set-mset X*) *R* **and**
tot: *totalp-on* (*set-mset X*) *R*
begin

16.1 Conversions

lemma *is-least-in-mset-wrt-iff*:
is-least-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#x\#\}.\ R\ x\ y)$
using *is-least-in-mset-wrt-def*[*OF trans asym tot*] **.**

lemma *is-greatest-in-mset-wrt-iff*:
is-greatest-in-mset-wrt $R\ X\ x \longleftrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#x\#\}.\ R\ y\ x)$
using *is-greatest-in-mset-wrt-def*[*OF trans asym tot*] **.**

lemma *is-minimal-in-mset-wrt-if-is-least-in-mset-wrt*[*intro*]:
is-least-in-mset-wrt $R\ X\ x \Longrightarrow$ *is-minimal-in-mset-wrt* $R\ X\ x$
unfolding *is-minimal-in-mset-wrt-iff*[*OF trans asym*]
unfolding *is-least-in-mset-wrt-def*[*OF trans asym tot*]
by (*metis add-mset-remove-trivial-eq asym asymp-onD insert-noteq-member*)

lemma *is-maximal-in-mset-wrt-if-is-greatest-in-mset-wrt*[*intro*]:
is-greatest-in-mset-wrt $R\ X\ x \Longrightarrow$ *is-maximal-in-mset-wrt* $R\ X\ x$
unfolding *is-maximal-in-mset-wrt-iff*[*OF trans asym*]
unfolding *is-greatest-in-mset-wrt-def*[*OF trans asym tot*]
by (*metis add-mset-remove-trivial-eq asym asymp-onD insert-noteq-member*)

lemma *is-strictly-minimal-in-mset-wrt-iff-is-least-in-mset-wrt*:
is-strictly-minimal-in-mset-wrt $R\ X =$ *is-least-in-mset-wrt* $R\ X$
unfolding *is-strictly-minimal-in-mset-wrt-iff*[*OF trans asym*] *is-least-in-mset-wrt-iff*

proof(*intro ext iffI*)
fix x
show $x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.\ \neg\ R^{==}\ y\ x) \Longrightarrow x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#x\#\}.\ R\ x\ y)$
by (*metis* (*mono-tags*, *lifting*) *in-diffD sup2CI tot totalp-onD*)
next
fix x
show $x \in\#\ X \wedge (\forall\, y \in\#\ X - \{\#x\#\}.\ R\ x\ y) \Longrightarrow x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.\ \neg\ R^{==}\ y\ x)$
by (*metis* (*full-types*) *asym asymp-onD in-diffD sup2E*)
qed

lemma *is-strictly-maximal-in-mset-wrt-iff-is-greatest-in-mset-wrt*:
is-strictly-maximal-in-mset-wrt $R\ X =$ *is-greatest-in-mset-wrt* $R\ X$
unfolding *is-strictly-maximal-in-mset-wrt-iff*[*OF trans asym*] *is-greatest-in-mset-wrt-iff*

proof(*intro ext iffI*)
fix x
show $x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.\ \neg\ R^{==}\ x\ y) \Longrightarrow x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.\ R\ y\ x)$
by (*metis* (*mono-tags*, *lifting*) *in-diffD sup2CI tot totalp-onD*)
next
fix x
show $x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.\ R\ y\ x) \Longrightarrow x \in\#\ X \wedge (\forall\, y{\in}\#X - \{\#x\#\}.$

$\neg R^{==} x y$)
  by (*metis* (*full-types*) *asym asymp-onD in-diffD sup2E*)
qed

16.2 Uniqueness

lemma *Uniq-is-minimal-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-minimal-in-mset-wrt R X x*
  unfolding *is-minimal-in-mset-wrt-iff* [*OF trans asym*]
  by (*smt* (*verit*, *best*) *Uniq-I tot totalp-onD*)

lemma *Uniq-is-maximal-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-maximal-in-mset-wrt R X x*
  unfolding *is-maximal-in-mset-wrt-iff* [*OF trans asym*]
  by (*smt* (*verit*, *best*) *Uniq-I tot totalp-onD*)

lemma *Uniq-is-least-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-least-in-mset-wrt R X x*
  using *is-least-in-mset-wrt-iff*
  by (*smt* (*verit*, *best*) *Uniq-I asym asymp-onD insert-DiffM insert-noteq-member*)

lemma *Uniq-is-greatest-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-greatest-in-mset-wrt R X x*
  unfolding *is-greatest-in-mset-wrt-iff*
  by (*smt* (*verit*, *best*) *Uniq-I asym asymp-onD insert-DiffM insert-noteq-member*)

lemma *Uniq-is-strictly-minimal-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-strictly-minimal-in-mset-wrt R X x*
  using *Uniq-is-least-in-mset-wrt*
  unfolding *is-strictly-minimal-in-mset-wrt-iff-is-least-in-mset-wrt*.

lemma *Uniq-is-strictly-maximal-in-mset-wrt*[*intro*]:
  $\exists_{\leq 1} x$. *is-strictly-maximal-in-mset-wrt R X x*
  using *Uniq-is-greatest-in-mset-wrt*
  unfolding *is-strictly-maximal-in-mset-wrt-iff-is-greatest-in-mset-wrt*.

16.3 Miscellaneous

lemma *is-least-in-mset-wrt-iff-is-minimal-and-count-eq-one*:
  is-least-in-mset-wrt R X x $\longleftrightarrow$ *is-minimal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
proof (*rule iffI*)
  assume *is-least-in-mset-wrt R X x*
  thus *is-minimal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
    unfolding *is-least-in-mset-wrt-iff is-minimal-in-mset-wrt-iff* [*OF trans asym*]
   by (*metis One-nat-def add-mset-remove-trivial-eq asym asymp-onD count-add-mset not-in-iff*)
next
  assume *is-minimal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
  then show *is-least-in-mset-wrt R X x*
    unfolding *is-least-in-mset-wrt-iff is-minimal-in-mset-wrt-iff* [*OF trans asym*]

by (*metis count-single in-diffD in-diff-count nat-less-le tot totalp-onD*)
qed

lemma *is-greatest-in-mset-wrt-iff-is-maximal-and-count-eq-one*:
is-greatest-in-mset-wrt R X x $\longleftrightarrow$ *is-maximal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
proof (*rule iffI*)
assume *is-greatest-in-mset-wrt R X x*
thus *is-maximal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
unfolding *is-greatest-in-mset-wrt-iff is-maximal-in-mset-wrt-iff*[*OF trans asym*]
by (*metis One-nat-def add-mset-remove-trivial-eq asym asymp-onD count-add-mset not-in-iff*)
next
assume *is-maximal-in-mset-wrt R X x* $\wedge$ *count X x* = *1*
then show *is-greatest-in-mset-wrt R X x*
unfolding *is-greatest-in-mset-wrt-iff is-maximal-in-mset-wrt-iff*[*OF trans asym*]
by (*metis count-single in-diffD in-diff-count nat-less-le tot totalp-onD*)
qed

lemma *count-ge-2-if-minimal-in-mset-wrt-and-not-least-in-mset-wrt*:
assumes *is-minimal-in-mset-wrt R X x* **and** $\neg$ *is-least-in-mset-wrt R X x*
shows *count X x* $\geq$ *2*
using *assms*
unfolding *is-least-in-mset-wrt-iff-is-minimal-and-count-eq-one*
by (*metis Suc-1 Suc-le-eq antisym-conv1 asym count-greater-eq-one-iff is-minimal-in-mset-wrt-iff trans*)

lemma *count-ge-2-if-maximal-in-mset-wrt-and-not-greatest-in-mset-wrt*:
assumes *is-maximal-in-mset-wrt R X x* **and** $\neg$ *is-greatest-in-mset-wrt R X x*
shows *count X x* $\geq$ *2*
using *assms*
unfolding *is-greatest-in-mset-wrt-iff-is-maximal-and-count-eq-one*
by (*metis Suc-1 Suc-le-eq antisym-conv1 asym count-greater-eq-one-iff is-maximal-in-mset-wrt-iff trans*)

lemma *explode-greatest-in-mset-wrt*:
assumes *max*: *is-greatest-in-mset-wrt R X x*
shows *add-mset x* {#*y* ∈# *X*. *y* ≠ *x*#} = *X*
using *max*[*folded is-strictly-maximal-in-mset-wrt-iff-is-greatest-in-mset-wrt*]
using *explode-strictly-maximal-in-mset-wrt*[*OF trans asym*]
by *metis*

end

lemma $multp_{HO}$*-if-maximal-wrt-less-that-maximal-wrt*:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**

tot: *totalp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
x1-maximal: *is-maximal-in-mset-wrt R M1 x1* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x2* **and**
R x1 x2
shows $multp_{HO}$ *R M1 M2*
proof −
have
trans1: *transp-on* (*set-mset M1*) *R* **and** *trans2*: *transp-on* (*set-mset M2*) *R* **and**
asym1: *asymp-on* (*set-mset M1*) *R* **and** *asym2*: *asymp-on* (*set-mset M2*) *R* **and**
tot1: *totalp-on* (*set-mset M1*) *R* **and** *tot2*: *totalp-on* (*set-mset M2*) *R*
using *trans*[*THEN transp-on-subset*] *asym*[*THEN asymp-on-subset*] *tot*[*THEN totalp-on-subset*]
by *simp-all*

have *x1-in*: *x1* ∈# *M1* **and** *x1-gr*: ∀ *y*∈#*M1*. *y* ≠ *x1* ⟶ ¬ *R x1 y*
using *x1-maximal*[*unfolded is-maximal-in-mset-wrt-iff*[*OF trans1 asym1*]] **by** *argo*+

have *x2-in*: *x2* ∈# *M2* **and** *x2-gr*: ∀ *y*∈#*M2*. *y* ≠ *x2* ⟶ ¬ *R x2 y*
using *x2-maximal*[*unfolded is-maximal-in-mset-wrt-iff*[*OF trans2 asym2*]] **by** *argo*+

show $multp_{HO}$ *R M1 M2*
unfolding $multp_{HO}$*-def*
proof (*intro conjI*)
show *M1* ≠ *M2*
using *x1-in x2-in x1-gr*
by (*metis* ‹*R x1 x2*› *asym2 asymp-onD*)
next
show ∀ *y*. *count M2 y* < *count M1 y* ⟶ (∃ *x*. *R y x* ∧ *count M1 x* < *count M2 x*)
using *x1-in x2-in x1-gr x2-gr*
by (*smt* (*verit*, *best*) *assms*(*6*) *asym1 asymp-onD count-greater-zero-iff count-inI dual-order.strict-trans local.trans subsetD sup-ge1 sup-ge2 tot1 totalp-onD transp-onD*)
qed
qed

lemma $multp_{DM}$*-if-maximal-wrt-less-that-maximal-wrt*:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
tot: *totalp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
x1-maximal: *is-maximal-in-mset-wrt R M1 x1* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x2* **and**
R x1 x2
shows $multp_{DM}$ *R M1 M2*
using $multp_{HO}$*-if-maximal-wrt-less-that-maximal-wrt*[*OF assms*, *THEN* $multp_{HO}$*-imp-*$multp_{DM}$]

.

lemma *multp-if-maximal-wrt-less-that-maximal-wrt*:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
tot: *totalp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
x1-maximal: *is-maximal-in-mset-wrt R M1 x1* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x2* **and**
R x1 x2
shows *multp R M1 M2*
using $multp_{DM}$*-if-maximal-wrt-less-that-maximal-wrt*[*OF assms*, *THEN* $multp_{DM}$*-imp-multp*]
.

lemma $multp_{HO}$*-if-same-maximal-wrt-and-count-lt*:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
tot: *totalp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
x1-maximal: *is-maximal-in-mset-wrt R M1 x* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x* **and**
count M1 x < *count M2 x*
shows $multp_{HO}$ *R M1 M2*
by (*metis assms*(*6*) *asym asymp-on-subset count-inI is-greatest-in-set-wrt-iff*
is-maximal-in-mset-wrt-def is-maximal-in-set-wrt-eq-is-greatest-in-set-wrt less-zeroE
local.trans $multp_{HO}$*-def order-less-imp-not-less sup-ge1 tot totalp-on-subset*
transp-on-subset
x1-maximal)

lemma *multp-if-same-maximal-wrt-and-count-lt*:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
tot: *totalp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
x1-maximal: *is-maximal-in-mset-wrt R M1 x* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x* **and**
count M1 x < *count M2 x*
shows *multp R M1 M2*
using $multp_{HO}$*-if-same-maximal-wrt-and-count-lt*[
OF assms, *THEN* $multp_{HO}$*-imp-*$multp_{DM}$, *THEN* $multp_{DM}$*-imp-multp*] .

lemma *less-than-maximal-wrt-if-*$multp_{HO}$:
assumes
trans: *transp-on* (*set-mset M1* ∪ *set-mset M2*) *R* **and**
asym: *asymp-on* (*set-mset M2*) *R* **and**
tot: *totalp-on* (*set-mset M2*) *R* **and**
x2-maximal: *is-maximal-in-mset-wrt R M2 x2* **and**
$multp_{HO}$ *R M1 M2* **and**

x1 $\in\#$ *M1*
shows $R^{==}$ *x1 x2*
proof −
have
trans2: *transp-on* (*set-mset M2*) *R* **and**
asym2: *asymp-on* (*set-mset M2*) *R* **and**
tot2: *totalp-on* (*set-mset M2*) *R*
using *trans*[*THEN transp-on-subset*] *asym*[*THEN asymp-on-subset*] *tot*[*THEN totalp-on-subset*]
by *simp-all*

have *x2-in*: *x2* $\in\#$ *M2* **and** *x2-gr*: $\forall$ *y*$\in\#$*M2*. *y* $\neq$ *x2* $\longrightarrow$ $\neg$ *R x2 y*
using *x2-maximal*[*unfolded is-maximal-in-mset-wrt-iff*[*OF trans2 asym2*]] **by** *argo+*

show *?thesis*
proof (*cases x1* $\in\#$ *M2*)
case *True*
thus *?thesis*
using *x2-gr* **by** (*metis* (*mono-tags*) *sup2CI tot2 totalp-onD x2-in*)
next
case *False*

hence *x1* $\in\#$ *M1* − *M2*
using ‹*x1* $\in\#$ *M1*› **by** (*simp add*: *in-diff-count not-in-iff*)

moreover have $\forall$ *k*$\in\#$*M1* − *M2*. $\exists$ *x*$\in\#$*M2* − *M1*. *R k x*
using $multp_{HO}$*-implies-one-step-strong*(*2*)[*OF* ‹$multp_{HO}$ *R M1 M2*›] **.**

ultimately obtain *x* **where** *x* $\in\#$ *M2* − *M1* **and** *R x1 x*
by *metis*

hence *x* $\neq$ *x2* $\longrightarrow$ $\neg$ *R x2 x*
using *x2-gr* **by** (*metis in-diffD*)

hence *x* $\neq$ *x2* $\longrightarrow$ *R x x2*
by (*metis* ‹*x* $\in\#$ *M2* − *M1*› *in-diffD tot2 totalp-onD x2-in*)

thus *?thesis*
using ‹*R x1 x*›
by (*meson Un-iff* ‹*x* $\in\#$ *M2* − *M1*› *assms*(*6*) *in-diffD local.trans sup2I1 transp-onD x2-in*)
qed
qed

17 Examples of duplicate handling in set and multiset definitions

lemma
  fixes *M* :: *nat multiset*
  defines $M \equiv \{\#0,\ 0,\ 1,\ 1,\ 2,\ 2\#\}$
  shows
    is-minimal-in-set-wrt (<) (*set-mset M*) *0*
    is-minimal-in-mset-wrt (<) *M 0*
    is-least-in-set-wrt (<) (*set-mset M*) *0*
    $\nexists$*y. is-least-in-mset-wrt* (<) *M y*
  by (*auto simp*: *M-def is-minimal-in-set-wrt-iff is-minimal-in-mset-wrt-def*
      is-least-in-set-wrt-iff is-least-in-mset-wrt-def)

lemma
  fixes *x y* :: *'a* **and** *M* :: *'a multiset*
  defines $M \equiv \{\#x,\ y,\ y\#\}$
  defines $R \equiv \lambda$- -. *False*
  assumes $x \neq y$
  shows
    is-maximal-in-mset-wrt R M x
    is-maximal-in-mset-wrt R M y
    is-strictly-maximal-in-mset-wrt R M x
    ¬ *is-strictly-maximal-in-mset-wrt R M y*
proof −
  have *transp-on-False*[*simp*]: $\bigwedge$*A. transp-on A* (λ- -. *False*)
    by (*simp add*: *transp-onI*)

  have *asymp-on-False*[*simp*]: $\bigwedge$*A. asymp-on A* (λ- -. *False*)
    by (*simp add*: *asymp-onI*)

  show
    is-maximal-in-mset-wrt R M x
    is-maximal-in-mset-wrt R M y
    is-strictly-maximal-in-mset-wrt R M x
    ¬ *is-strictly-maximal-in-mset-wrt R M y*
   unfolding *is-maximal-in-mset-wrt-iff*[*of M R*, *unfolded R-def*, *simplified*, *folded R-def*]
    unfolding *is-strictly-maximal-in-mset-wrt-iff*[*of M R*, *unfolded R-def*, *simplified*, *folded R-def*]
    unfolding *atomize-conj*
    using ‹$x \neq y$›
    by (*simp add*: *M-def*)
qed

18 Hide stuff

We restrict the public interface to ease future internal changes.

hide-fact *is-minimal-in-mset-wrt-def is-maximal-in-mset-wrt-def*
hide-fact *is-strictly-minimal-in-mset-wrt-def is-strictly-maximal-in-mset-wrt-def*
hide-fact *is-least-in-mset-wrt-def is-greatest-in-mset-wrt-def*

19 Integration in type classes

abbreviation (**in** *order*) *is-minimal-in-mset* **where**
is-minimal-in-mset ≡ *is-minimal-in-mset-wrt* (<)

abbreviation (**in** *order*) *is-maximal-in-mset* **where**
is-maximal-in-mset ≡ *is-maximal-in-mset-wrt* (<)

abbreviation (**in** *order*) *is-strictly-minimal-in-mset* **where**
is-strictly-minimal-in-mset ≡ *is-strictly-minimal-in-mset-wrt* (<)

abbreviation (**in** *order*) *is-strictly-maximal-in-mset* **where**
is-strictly-maximal-in-mset ≡ *is-strictly-maximal-in-mset-wrt* (<)

lemmas (**in** *order*) *is-minimal-in-mset-iff* =
is-minimal-in-mset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-maximal-in-mset-iff* =
is-maximal-in-mset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-strictly-minimal-in-mset-iff* =
is-strictly-minimal-in-mset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-strictly-maximal-in-mset-iff* =
is-strictly-maximal-in-mset-wrt-iff[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-minimal-in-mset-if-is-strictly-minimal-in-mset*[*intro*] =
is-minimal-in-mset-wrt-if-is-strictly-minimal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-maximal-in-mset-if-is-strictly-maximal-in-mset*[*intro*] =
is-maximal-in-mset-wrt-if-is-strictly-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-minimal-in-mset* =
ex-minimal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *ex-maximal-in-mset* =
ex-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *explode-maximal-in-mset* =
explode-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *explode-strictly-maximal-in-mset* =
explode-strictly-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less*]

lemmas (**in** *order*) *is-minimal-in-filter-mset-iff* =

is-minimal-in-filter-mset-wrt-iff[*OF transp-on-less asymp-on-less*]

abbreviation (**in** *linorder*) *is-least-in-mset* **where**
is-least-in-mset ≡ *is-least-in-mset-wrt* (<)

abbreviation (**in** *linorder*) *is-greatest-in-mset* **where**
is-greatest-in-mset ≡ *is-greatest-in-mset-wrt* (<)

lemmas (**in** *linorder*) *is-least-in-mset-iff* =
is-least-in-mset-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-greatest-in-mset-iff* =
is-greatest-in-mset-wrt-iff[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-minimal-in-mset-if-is-least-in-mset*[*intro*] =
is-minimal-in-mset-wrt-if-is-least-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-maximal-in-mset-if-is-greatest-in-mset*[*intro*] =
is-maximal-in-mset-wrt-if-is-greatest-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-minimal-in-mset*[*intro*] =
Uniq-is-minimal-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-maximal-in-mset*[*intro*] =
Uniq-is-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-least-in-mset*[*intro*] =
Uniq-is-least-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-greatest-in-mset*[*intro*] =
Uniq-is-greatest-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-strictly-minimal-in-mset*[*intro*] =
Uniq-is-strictly-minimal-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *Uniq-is-strictly-maximal-in-mset*[*intro*] =
Uniq-is-strictly-maximal-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-least-in-mset-iff-is-minimal-and-count-eq-one* =
is-least-in-mset-wrt-iff-is-minimal-and-count-eq-one[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *is-greatest-in-mset-iff-is-maximal-and-count-eq-one* =
is-greatest-in-mset-wrt-iff-is-maximal-and-count-eq-one[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *count-ge-2-if-minimal-in-mset-and-not-least-in-mset* =
count-ge-2-if-minimal-in-mset-wrt-and-not-least-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *count-ge-2-if-maximal-in-mset-and-not-greatest-in-mset* =
count-ge-2-if-maximal-in-mset-wrt-and-not-greatest-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *explode-greatest-in-mset* =
explode-greatest-in-mset-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) $multp_{HO}$*-if-maximal-less-that-maximal* =
$multp_{HO}$*-if-maximal-wrt-less-that-maximal-wrt*[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) $multp_{DM}$*-if-maximal-less-that-maximal* =
$multp_{DM}$*-if-maximal-wrt-less-that-maximal-wrt*[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *multp-if-maximal-less-that-maximal* =
multp-if-maximal-wrt-less-that-maximal-wrt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) $multp_{HO}$*-if-same-maximal-and-count-lt* =
$multp_{HO}$*-if-same-maximal-wrt-and-count-lt*[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *multp-if-same-maximal-and-count-lt* =
multp-if-same-maximal-wrt-and-count-lt[*OF transp-on-less asymp-on-less totalp-on-less*]

lemmas (**in** *linorder*) *less-than-maximal-if-*$multp_{HO}$ =
less-than-maximal-wrt-if-$multp_{HO}$[*OF transp-on-less asymp-on-less totalp-on-less*]

lemma (**in** *linorder*)
assumes*is-greatest-in-mset C L*
shows $C - \{\#L\#\} = \{\#K \in\# C.\ K \neq L\#\}$
using *assms*
by (*smt* (*verit*, *del-insts*) *add-diff-cancel-left′ diff-subset-eq-self diff-zero filter-empty-mset filter-mset-add-mset filter-mset-eq-conv insert-DiffM2 local.is-greatest-in-mset-iff local.not-less-iff-gr-or-eq*)

end