Formalization of Nested Multisets, Hereditary Multisets, and Syntactic Ordinals

Jasmin Christian Blanchette, Mathias Fleury, and Dmitriy Traytel

December 17, 2016

Abstract

This Isabelle/HOL formalization introduces a nested multiset datatype and defines Dershowitz and Manna's nested multiset order. The order is proved well founded and linear. By removing one constructor, we transform the nested multisets into hereditary multisets. These are isomorphic to the syntactic ordinals—the ordinals can be recursively expressed in Cantor normal form. Addition, subtraction, multiplication, and linear orders are provided on this type.

Contents

1 Introduction

This Isabelle/HOL formalization introduces a nested multiset datatype and defines Dershowitz and Manna's nested multiset order. The order is proved well founded and linear. By removing one constructor, we transform the nested multisets into hereditary multisets. These are isomorphic to the syntactic ordinals—the ordinals can be recursively expressed in Cantor normal form. Addition, subtraction, multiplication, and linear orders are provided on this type.

In addition, signed (or hybrid) multisets are provided (i.e., multisets with potentially negative multiplicities), as well as signed hereditary multisets and signed ordinals (e.g., $\omega^2 - 2\omega + 1$).

2 More about Multisets

theory *Multiset_More*
imports ~~/*src/HOL/Library/Multiset_Order*
begin

Isabelle's theory of finite multisets is not as developed as other areas, such as lists and sets. The present theory introduces some missing concepts and lemmas. Some of it is expected to move to Isabelle's library.

2.1 Basic Setup

declare
diff_single_trivial [*simp*]
in_image_mset [*iff*]
image_mset.compositionality [*simp*]

mset_subset_eqD[*dest*, *intro?*]

Multiset.in_multiset_in_set[*simp*]
inter_add_left1[*simp*]
inter_add_left2[*simp*]
inter_add_right1[*simp*]
inter_add_right2[*simp*]

sum_mset_sum_list[*simp*]

lemma *add_mset_in_multiset'*:
assumes *M*: $M \in multiset$
shows $(\lambda b.\ if\ b = a\ then\ Suc\ (M\ a)\ else\ M\ b) \in multiset$
⟨*proof*⟩

2.2 Lemmas about the Multiset Order

instantiation *multiset* :: (*linorder*) *distrib_lattice*
begin

definition *inf_multiset* :: $'a\ multiset \Rightarrow 'a\ multiset \Rightarrow 'a\ multiset$ **where**
$inf\_multiset\ A\ B = (if\ A < B\ then\ A\ else\ B)$

definition *sup_multiset* :: $'a\ multiset \Rightarrow 'a\ multiset \Rightarrow 'a\ multiset$ **where**
$sup\_multiset\ A\ B = (if\ B > A\ then\ B\ else\ A)$

instance
⟨*proof*⟩

end

lemma *all_lt_Max_imp_lt_multiset*: $N \neq \{\#\} \Longrightarrow (\forall a \in\# M.\ a < Max\ (set\_mset\ N)) \Longrightarrow M < N$
⟨*proof*⟩

lemma *lt_imp_ex_count_lt*: $M < N \Longrightarrow \exists y.\ count\ M\ y < count\ N\ y$
⟨*proof*⟩

lemma *subset_imp_less_multiset*:
fixes *A*
assumes *a_sub_b*: $A \subset\# B$
shows $A < B$
⟨*proof*⟩

lemma *image_mset_mono*:
assumes
mono_f: $\forall x \in set\_mset\ M.\ \forall y \in set\_mset\ N.\ x < y \longrightarrow f\ x < f\ y$ **and**
less: $M < N$
shows $image\_mset\ f\ M < image\_mset\ f\ N$
⟨*proof*⟩

2.3 Lemmas about Intersection, Union and Pointwise Inclusion

lemma *mset_inter_single*: $x \in\# \Sigma \Longrightarrow \Sigma \cap\# \{\#x\#\} = \{\#x\#\}$
⟨*proof*⟩

lemma *subset_add_mset_notin_subset_mset*: ‹$A \subseteq\# add\_mset\ b\ B \Longrightarrow b \notin\# A \Longrightarrow A \subseteq\# B$›
⟨*proof*⟩

psubsetE is the set counterpart

lemma *subset_msetE* [*elim*!]:
$[\![A \subset\# B;\ [\![A \subseteq\# B;\ \sim (B \subseteq\# A)]\!] \Longrightarrow R]\!] \Longrightarrow R$
⟨*proof*⟩

lemma *Diff_triv_mset*: $M \cap\# N = \{\#\} \Longrightarrow M - N = M$
⟨*proof*⟩

lemma *diff_intersect_sym_diff*: $(A - B) \cap\# (B - A) = \{\#\}$
⟨*proof*⟩

2.4 Lemmas about Size

This sections adds various lemmas about size. Most lemmas have a finite set equivalent.

lemma *size_mset_SucE*: $size\ A = Suc\ n \Longrightarrow (\bigwedge a\ B.\ A = \{\#a\#\} + B \Longrightarrow size\ B = n \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

lemma *size_Suc_Diff1*: $x \in\# \Sigma \Longrightarrow Suc\ (size\ (\Sigma - \{\#x\#\})) = size\ \Sigma$
⟨*proof*⟩

lemma *size_Diff_singleton*: $x \in\# \Sigma \Longrightarrow size\ (\Sigma - \{\#x\#\}) = size\ \Sigma - 1$
⟨*proof*⟩

lemma *size_Diff_singleton_if*: $size\ (A - \{\#x\#\}) = (if\ x \in\# A\ then\ size\ A - 1\ else\ size\ A)$
⟨*proof*⟩

lemma *size_Un_Int*: $size\ A + size\ B = size\ (A \cup\# B) + size\ (A \cap\# B)$
⟨*proof*⟩

lemma *size_Un_disjoint*:
assumes $A \cap\# B = \{\#\}$
shows $size\ (A \cup\# B) = size\ A + size\ B$
⟨*proof*⟩

lemma *size_Diff_subset_Int*:
shows $size\ (\Sigma - \Sigma') = size\ \Sigma - size\ (\Sigma \cap\# \Sigma')$
⟨*proof*⟩

lemma *diff_size_le_size_Diff*: $size\ (\Sigma::\ \_\ multiset) - size\ \Sigma' \leq size\ (\Sigma - \Sigma')$
⟨*proof*⟩

lemma *size_Diff1_less*: $x \in\# \Sigma \Longrightarrow size\ (\Sigma - \{\#x\#\}) < size\ \Sigma$
⟨*proof*⟩

lemma *size_Diff2_less*: $x \in\# \Sigma \Longrightarrow y \in\# \Sigma \Longrightarrow size\ (\Sigma - \{\#x\#\} - \{\#y\#\}) < size\ \Sigma$
⟨*proof*⟩

lemma *size_Diff1_le*: $size\ (\Sigma - \{\#x\#\}) \leq size\ \Sigma$
⟨*proof*⟩

lemma *size_psubset*: $\Sigma \subseteq\# \Sigma' \Longrightarrow size\ \Sigma < size\ \Sigma' \Longrightarrow \Sigma \subset\# \Sigma'$
⟨*proof*⟩

2.5 Lemmas about Filter and Image

lemma *count_image_mset_ge_count*:
$count\ (image\_mset\ f\ A)\ (f\ b) \geq count\ A\ b$
⟨*proof*⟩

lemma *count_image_mset_inj*:
assumes ‹*inj f*›
shows ‹$count\ (image\_mset\ f\ M)\ (f\ x) = count\ M\ x$›
⟨*proof*⟩

lemma *mset_filter_compl*: $mset\ (filter\ p\ xs) + mset\ (filter\ (Not \circ p)\ xs) = mset\ xs$
⟨*proof*⟩

lemma *comprehension_mset_False*[*simp*]: $\{\#\ L \in\#\ A.\ False\#\} = \{\#\}$

⟨*proof*⟩

Near duplicate of *filter_eq_replicate_mset*: $\{\#y \in\# ?D.\ y = ?x\#\} = replicate\_mset\ (count\ ?D\ ?x)\ ?x$.

lemma *filter_mset_eq*: $filter\_mset\ (op = L)\ A = replicate\_mset\ (count\ A\ L)\ L$
⟨*proof*⟩

See *filter_cong* for the set version. Mark as [*fundef_cong*] too?

lemma *filter_mset_cong*:
assumes [*simp*]: $M = M'$ **and** [*simp*]: $\bigwedge a.\ a \in\# M \Longrightarrow P\ a = Q\ a$
shows $filter\_mset\ P\ M = filter\_mset\ Q\ M$
⟨*proof*⟩

lemma *filter_mset_filter_mset*: $filter\_mset\ Q\ (filter\_mset\ P\ M) = \{\#x \in\# M.\ P\ x \wedge Q\ x\#\}$
⟨*proof*⟩

lemma *image_mset_const*: $\{\#c.\ x \in\# M\#\} = replicate\_mset\ (size\ M)\ c$
⟨*proof*⟩

lemma *image_mset_filter_swap*: $image\_mset\ f\ \{\#\ x \in\# M.\ P\ (f\ x)\#\} = \{\#\ x \in\# image\_mset\ f\ M.\ P\ x\#\}$
⟨*proof*⟩

lemma *image_mset_cong2*[*cong*]:
$(\bigwedge x.\ x \in\# M \Longrightarrow f\ x = g\ x) \Longrightarrow M = N \Longrightarrow image\_mset\ f\ M = image\_mset\ g\ N$
⟨*proof*⟩

lemma *filter_mset_empty_conv*: ‹$(filter\_mset\ P\ M = \{\#\}) = (\forall L \in\# M.\ \neg\ P\ L)$›
⟨*proof*⟩

lemma *multiset_filter_mono2*: ‹$filter\_mset\ P\ A \subseteq\# filter\_mset\ Q\ A \longleftrightarrow$
$(\forall a \in\# A.\ P\ a \longrightarrow Q\ a)$›
⟨*proof*⟩

lemma *image_filter_cong*:
assumes ‹$\bigwedge C.\ C \in\# M \Longrightarrow P\ C \Longrightarrow f\ C = g\ C$›
shows
‹$\{\#f\ C.\ C \in\# \{\#C \in\# M.\ P\ C\#\}\#\} = \{\#g\ C | C \in\# M.\ P\ C\#\}$›
⟨*proof*⟩

lemma *image_mset_filter_swap2*: ‹$\{\#C \in\# \{\#P\ x.\ x \in\# D\#\}.\ Q\ C\ \#\} = \{\#P\ x.\ x \in\# \{\#C |\ C \in\# D.\ Q\ (P\ C)\#\}\#\}$›
⟨*proof*⟩

2.6 Lemma about Sum

lemma *count_sum_mset_if_1_0*: ‹$count\ M\ a = (\sum x \in\# M.\ if\ x = a\ then\ 1\ else\ 0)$›
⟨*proof*⟩

2.7 Lemmas about Remove

lemma *set_mset_minus_replicate_mset*[*simp*]:
$n \geq count\ A\ a \Longrightarrow set\_mset\ (A - replicate\_mset\ n\ a) = set\_mset\ A - \{a\}$
$n < count\ A\ a \Longrightarrow set\_mset\ (A - replicate\_mset\ n\ a) = set\_mset\ A$
⟨*proof*⟩

abbreviation *removeAll_mset* :: $'a \Rightarrow 'a\ multiset \Rightarrow 'a\ multiset$ **where**
$removeAll\_mset\ C\ M \equiv M - replicate\_mset\ (count\ M\ C)\ C$

lemma *mset_removeAll*[*simp*, *code*]:
$removeAll\_mset\ C\ (mset\ L) = mset\ (removeAll\ C\ L)$
⟨*proof*⟩

lemma *removeAll_mset_filter_mset*:
$removeAll\_mset\ C\ M = filter\_mset\ (op \neq C)\ M$

⟨*proof*⟩

abbreviation *remove1_mset* :: *'a* ⇒ *'a multiset* ⇒ *'a multiset* **where**
remove1_mset C M ≡ *M* − {#*C*#}

lemma *in_remove1_mset_neq*:
assumes *ab*: *a* ≠ *b*
shows *a* ∈# *remove1_mset b C* ⟷ *a* ∈# *C*
⟨*proof*⟩

lemma *size_mset_removeAll_mset_le_iff*: *size* (*removeAll_mset x M*) < *size M* ⟷ *x* ∈# *M*
⟨*proof*⟩

lemma *size_remove1_mset_If*: ‹*size* (*remove1_mset x M*) = *size M* − (*if x* ∈# *M then* 1 *else* 0)›
⟨*proof*⟩

lemma *size_mset_remove1_mset_le_iff*: *size* (*remove1_mset x M*) < *size M* ⟷ *x* ∈# *M*
⟨*proof*⟩

lemma *single_remove1_mset_eq*:
add_mset a (*remove1_mset a M*) = *M* ⟷ *a* ∈# *M*
⟨*proof*⟩

lemma *remove_1_mset_id_iff_notin*:
remove1_mset a M = *M* ⟷ *a* ∉# *M*
⟨*proof*⟩

lemma *id_remove_1_mset_iff_notin*:
M = *remove1_mset a M* ⟷ *a* ∉# *M*
⟨*proof*⟩

lemma *remove1_mset_eqE*:
remove1_mset L x1 = *M* ⟹
(*L* ∈# *x1* ⟹ *x1* = *M* + {#*L*#} ⟹ *P*) ⟹
(*L* ∉# *x1* ⟹ *x1* = *M* ⟹ *P*) ⟹
P
⟨*proof*⟩

lemma *image_filter_ne_mset*[*simp*]:
image_mset f {#*x* ∈# *M*. *f x* ≠ *y*#} = *removeAll_mset y* (*image_mset f M*)
⟨*proof*⟩

lemma *image_mset_remove1_mset_if*:
image_mset f (*remove1_mset a M*) =
(*if a* ∈# *M then remove1_mset* (*f a*) (*image_mset f M*) *else image_mset f M*)
⟨*proof*⟩

lemma *filter_mset_neq*: {#*x* ∈# *M*. *x* ≠ *y*#} = *removeAll_mset y M*
⟨*proof*⟩

lemma *filter_mset_neq_cond*: {#*x* ∈# *M*. *P x* ∧ *x* ≠ *y*#} = *removeAll_mset y* {# *x*∈#*M*. *P x*#}
⟨*proof*⟩

lemma *remove1_mset_add_mset_If*:
remove1_mset L (*add_mset L′ C*) = (*if L* = *L′ then C else remove1_mset L C* + {#*L′*#})
⟨*proof*⟩

lemma *minus_remove1_mset_if*:
A − *remove1_mset b B* = (*if b* ∈# *B* ∧ *b* ∈# *A* ∧ *count A b* ≥ *count B b then* {#*b*#} + (*A* − *B*) *else A* − *B*)
⟨*proof*⟩

lemma *add_mset_eq_add_mset_ne*:
a ≠ *b* ⟹ *add_mset a A* = *add_mset b B* ⟷ *a* ∈# *B* ∧ *b* ∈# *A* ∧ *A* = *add_mset b* (*B* − {#*a*#})

⟨*proof*⟩

This lemma allows to split equality like $\{\#K,\ K'\#\} = \{\#L,\ L'\#\}$ into $K = L \wedge L = L' \vee K \neq L \wedge K = L' \wedge K' = L$. It can lead to a explosion of the numbers of cases.

lemma *add_mset_eq_add_mset*: ‹*add_mset a M = add_mset b M'* ⟷
$(a = b \wedge M = M') \vee (a \neq b \wedge b \in\# M \wedge$ *add_mset a* $(M - \{\#b\#\}) = M')$›
⟨*proof*⟩

lemma *add_mset_remove_trivial_iff*: ‹$N =$ *add_mset a* $(N - \{\#b\#\}) \longleftrightarrow a \in\# N \wedge a = b$›
⟨*proof*⟩

lemma *trivial_add_mset_remove_iff*: ‹*add_mset a* $(N - \{\#b\#\}) = N \longleftrightarrow a \in\# N \wedge a = b$›
⟨*proof*⟩

lemma *remove1_single_empty_iff*[*simp*]: ‹*remove1_mset L* $\{\#L'\#\} = \{\#\} \longleftrightarrow L = L'$›
⟨*proof*⟩

2.8 Lemmas about Replicate

lemma *replicate_mset_minus_replicate_mset_same*[*simp*]:
replicate_mset m x − *replicate_mset n x* = *replicate_mset* $(m - n)$ *x*
⟨*proof*⟩

lemma *replicate_mset_subset_iff_lt*[*simp*]: *replicate_mset m x* $\subset\#$ *replicate_mset n x* $\longleftrightarrow m < n$
⟨*proof*⟩

lemma *replicate_mset_subseteq_iff_le*[*simp*]: *replicate_mset m x* $\subseteq\#$ *replicate_mset n x* $\longleftrightarrow m \leq n$
⟨*proof*⟩

lemma *replicate_mset_lt_iff_lt*[*simp*]: *replicate_mset m x* < *replicate_mset n x* $\longleftrightarrow m < n$
⟨*proof*⟩

lemma *replicate_mset_le_iff_le*[*simp*]: *replicate_mset m x* $\leq$ *replicate_mset n x* $\longleftrightarrow m \leq n$
⟨*proof*⟩

lemma *replicate_mset_eq_iff*[*simp*]:
replicate_mset m x = *replicate_mset n y* $\longleftrightarrow m = n \wedge (m \neq 0 \longrightarrow x = y)$
⟨*proof*⟩

lemma *replicate_mset_plus*: *replicate_mset* $(a + b)$ *C* = *replicate_mset a C* + *replicate_mset b C*
⟨*proof*⟩

lemma *mset_replicate_replicate_mset*: *mset* (*replicate n L*) = *replicate_mset n L*
⟨*proof*⟩

lemma *set_mset_single_iff_replicate_mset*:
set_mset U = $\{a\}$ ⟷ $(\exists n > 0.\ U =$ *replicate_mset n a*$)$ (**is** *?S* ⟷ *?R*)
⟨*proof*⟩

lemma *ex_replicate_mset_if_all_elems_eq*:
assumes $\forall x \in\# M.\ x = y$
shows $\exists n.\ M =$ *replicate_mset n y*
⟨*proof*⟩

2.9 Multiset and Set Conversions

lemma *count_mset_set_if*: *count* (*mset_set A*) *a* = (*if a* ∈ *A* ∧ *finite A then 1 else 0*)
⟨*proof*⟩

lemma *mset_set_set_mset_empty_mempty*[*iff*]: *mset_set* (*set_mset D*) = $\{\#\} \longleftrightarrow D = \{\#\}$
⟨*proof*⟩

lemma *count_mset_set_le_one*: *count* (*mset_set A*) *x* ≤ *1*
⟨*proof*⟩

lemma *mset_set_subseteq_mset_set*[*iff*]:
assumes *finite A finite B*
shows *mset_set A* ⊆# *mset_set B* ⟷ *A* ⊆ *B*
⟨*proof*⟩

lemma *mset_set_set_mset_subseteq*[*simp*]: *mset_set* (*set_mset A*) ⊆# *A*
⟨*proof*⟩

lemma *mset_sorted_list_of_set*[*simp*]: *mset* (*sorted_list_of_set A*) = *mset_set A*
⟨*proof*⟩

lemma *mset_take_subseteq*: *mset* (*take n xs*) ⊆# *mset xs*
⟨*proof*⟩

2.10 Duplicate Removal

definition *remdups_mset* :: *'v multiset* ⇒ *'v multiset* **where**
remdups_mset S = *mset_set* (*set_mset S*)

lemma *set_mset_remdups_mset*[*simp*]: ‹*set_mset* (*remdups_mset A*) = *set_mset A*›
⟨*proof*⟩

lemma *count_remdups_mset_eq_1*: *a* ∈# *remdups_mset A* ⟷ *count* (*remdups_mset A*) *a* = *1*
⟨*proof*⟩

lemma *remdups_mset_empty*[*simp*]: *remdups_mset* {#} = {#}
⟨*proof*⟩

lemma *remdups_mset_singleton*[*simp*]: *remdups_mset* {#*a*#} = {#*a*#}
⟨*proof*⟩

lemma *remdups_mset_eq_empty*[*iff*]: *remdups_mset D* = {#} ⟷ *D* = {#}
⟨*proof*⟩

lemma *remdups_mset_singleton_sum*[*simp*]:
remdups_mset (*add_mset a A*) = (*if a* ∈# *A then remdups_mset A else add_mset a* (*remdups_mset A*))
⟨*proof*⟩

lemma *mset_remdups_remdups_mset*[*simp*]: *mset* (*remdups D*) = *remdups_mset* (*mset D*)
⟨*proof*⟩

declare *mset_remdups_remdups_mset*[*symmetric*, *code*]

definition *distinct_mset* :: *'a multiset* ⇒ *bool* **where**
distinct_mset S ⟷ (∀ *a*. *a* ∈# *S* ⟶ *count S a* = *1*)

Another characterisation of *distinct_mset*:

lemma *distinct_mset_count_less_1*: *distinct_mset S* ⟷ (∀ *a*. *count S a* ≤ *1*)
⟨*proof*⟩

lemma *distinct_mset_empty*[*simp*]: *distinct_mset* {#}
⟨*proof*⟩

lemma *distinct_mset_singleton*: *distinct_mset* {#*a*#}
⟨*proof*⟩

lemma *distinct_mset_union*:
assumes *dist*: *distinct_mset* (*A* + *B*)
shows *distinct_mset A*
⟨*proof*⟩

lemma *distinct_mset_minus*[*simp*]:
distinct_mset A $\Longrightarrow$ *distinct_mset* (*A* − *B*)
⟨*proof*⟩

lemma *count_remdups_mset_If*: ‹*count* (*remdups_mset A*) *a* = (*if a* ∈# *A then 1 else 0*)›
⟨*proof*⟩

lemma *distinct_mset_rempdups_union_mset*:
assumes *distinct_mset A* **and** *distinct_mset B*
shows *A* ∪# *B* = *remdups_mset* (*A* + *B*)
⟨*proof*⟩

lemma *distinct_mset_add_mset*[*simp*]: *distinct_mset* (*add_mset a L*) ⟷ *a* ∉# *L* ∧ *distinct_mset L*
⟨*proof*⟩

lemma *distinct_mset_size_eq_card*: *distinct_mset C* $\Longrightarrow$ *size C* = *card* (*set_mset C*)
⟨*proof*⟩

lemma *distinct_mset_add*:
distinct_mset (*L* + *L'*) ⟷ *distinct_mset L* ∧ *distinct_mset L'* ∧ *L* ∩# *L'* = {#} (**is** *?A* ⟷ *?B*)
⟨*proof*⟩

lemma *distinct_mset_set_mset_ident*[*simp*]: *distinct_mset M* $\Longrightarrow$ *mset_set* (*set_mset M*) = *M*
⟨*proof*⟩

lemma *distinct_finite_set_mset_subseteq_iff*[*iff*]:
assumes *dist*: *distinct_mset M* **and** *fin*: *finite N*
shows *set_mset M* ⊆ *N* ⟷ *M* ⊆# *mset_set N*
⟨*proof*⟩

lemma *distinct_mem_diff_mset*:
assumes *dist*: *distinct_mset M* **and** *mem*: *x* ∈ *set_mset* (*M* − *N*)
shows *x* ∉ *set_mset N*
⟨*proof*⟩

lemma *distinct_set_mset_eq*:
assumes
dist_m: *distinct_mset M* **and**
dist_n: *distinct_mset N* **and**
set_eq: *set_mset M* = *set_mset N*
shows *M* = *N*
⟨*proof*⟩

lemma *distinct_mset_union_mset*[*simp*]:
‹*distinct_mset* (*D* ∪# *C*) ⟷ *distinct_mset D* ∧ *distinct_mset C*›
⟨*proof*⟩

lemma *distinct_mset_inter_mset*:
assumes
distinct_mset D **and**
distinct_mset C
shows *distinct_mset* (*D* ∩# *C*)
⟨*proof*⟩

lemma *distinct_mset_remove1_All*: *distinct_mset C* $\Longrightarrow$ *remove1_mset L C* = *removeAll_mset L C*
⟨*proof*⟩

lemma *distinct_mset_size_2*: *distinct_mset* {#*a*, *b*#} ⟷ *a* ≠ *b*
⟨*proof*⟩

lemma *distinct_mset_filter*: *distinct_mset M* $\Longrightarrow$ *distinct_mset* {# *L* ∈# *M*. *P L*#}
⟨*proof*⟩

lemma *distinct_mset_mset_distinct*[*simp*]: ‹*distinct_mset* (*mset xs*) = *distinct xs*›
⟨*proof*⟩

lemma *distinct_image_mset_inj*:
‹*inj_on f* (*set_mset M*) ⟹ *distinct_mset* (*image_mset f M*) ⟷ *distinct_mset M*›
⟨*proof*⟩

2.11 Repeat Operation

lemma *repeat_mset_compower*: *repeat_mset n A* = ((*op* + *A*) ^^ *n*) {#}
⟨*proof*⟩

lemma *repeat_mset_prod*: *repeat_mset* (*m* ∗ *n*) *A* = ((*op* + (*repeat_mset n A*)) ^^ *m*) {#}
⟨*proof*⟩

2.12 Cartesian Product

Definition of the cartesian products over multisets. The construction mimics of the cartesian product on sets and use the same theorem names (adding only the suffix *_mset* to Sigma and Times). See file `~~/src/HOL/Product_Type.thy`

definition *Sigma_mset* :: 'a *multiset* ⇒ ('a ⇒ 'b *multiset*) ⇒ ('a × 'b) *multiset* **where**
Sigma_mset A B ≡ ⋃# {#{#(*a*, *b*). *b* ∈# *B a*#}. *a* ∈# *A* #}

abbreviation *Times_mset* :: 'a *multiset* ⇒ 'b *multiset* ⇒ ('a × 'b) *multiset* (**infixr** ×*mset* 80) **where**
Times_mset A B ≡ *Sigma_mset A* (λ_. *B*)

hide-const (**open**) *Times_mset*

Contrary to the set version *A* × *B*, we use the non-ASCII symbol ∈#.

syntax
_Sigma_mset :: [*pttrn*, 'a *multiset*, 'b *multiset*] => ('a ∗ 'b) *multiset*
((*3SIGMAMSET* _∈#_./ _) [*0*, *0*, *10*] *10*)
translations
SIGMAMSET x∈#*A*. *B* == *CONST Sigma_mset A* (λ*x*. *B*)

Link between the multiset and the set cartesian product:

lemma *Times_mset_Times*: *set_mset* (*A* ×*mset B*) = *set_mset A* × *set_mset B*
⟨*proof*⟩

lemma *Sigma_msetI* [*intro*!]: ⟦*a* ∈# *A*; *b* ∈# *B a*⟧ ⟹ (*a*, *b*) ∈# *Sigma_mset A B*
⟨*proof*⟩

lemma *Sigma_msetE* [*elim*!]:
⟦ *c*∈# *Sigma_mset A B*;
⋀*x y*.⟦ *x*∈#*A*; *y*∈#*B*(*x*); *c*=(*x*,*y*) ⟧ ⟹ *P*
⟧ ⟹ *P*
— The general elimination rule.
⟨*proof*⟩

Elimination of (*a*, *b*) ∈# *A* ×*mset B* – introduces no eigenvariables.

lemma *Sigma_msetD1*: (*a*, *b*) ∈# *Sigma_mset A B* ⟹ *a* ∈# *A*
⟨*proof*⟩

lemma *Sigma_msetD2*: (*a*, *b*) ∈# *Sigma_mset A B* ⟹ *b* ∈# *B a*
⟨*proof*⟩

lemma *Sigma_msetE2*:
⟦ (*a*, *b*) ∈# *Sigma_mset A B*;
⟦ *a*∈#*A*; *b*∈#*B*(*a*) ⟧ ⟹ *P*
⟧ ⟹ *P*
⟨*proof*⟩

lemma *Sigma_mset_cong*:
$[\![A = B;\ \bigwedge x.\ x \in\# B \Longrightarrow C\ x = D\ x]\!]$
$\Longrightarrow (SIGMAMSET\ x{\in}\#A.\ C\ x) = (SIGMAMSET\ x{\in}\#\ B.\ D\ x)$
⟨*proof*⟩

lemma *count_sum_mset*: $count\ (\bigcup\#M)\ b = (\sum P{\in}\#M.\ count\ P\ b)$
⟨*proof*⟩

lemma *Sigma_mset_plus_distrib1*[*simp*]: $Sigma\_mset\ (A + B)\ C = Sigma\_mset\ A\ C + Sigma\_mset\ B\ C$
⟨*proof*⟩

lemma *Sigma_mset_plus_distrib2*[*simp*]:
$Sigma\_mset\ A\ (\lambda i.\ B\ i + C\ i) = Sigma\_mset\ A\ B + Sigma\_mset\ A\ C$
⟨*proof*⟩

lemma *Times_mset_single_left*: $\{\#a\#\} \times mset\ B = image\_mset\ (Pair\ a)\ B$
⟨*proof*⟩

lemma *Times_mset_single_right*: $A \times mset\ \{\#b\#\} = image\_mset\ (\lambda a.\ Pair\ a\ b)\ A$
⟨*proof*⟩

lemma *Times_mset_single_single*[*simp*]: $\{\#a\#\} \times mset\ \{\#b\#\} = \{\#(a,\ b)\#\}$
⟨*proof*⟩

lemma *count_image_mset_Pair*:
$count\ (image\_mset\ (Pair\ a)\ B)\ (x,\ b) = (if\ x = a\ then\ count\ B\ b\ else\ 0)$
⟨*proof*⟩

lemma *count_Sigma_mset*: $count\ (Sigma\_mset\ A\ B)\ (a,\ b) = count\ A\ a * count\ (B\ a)\ b$
⟨*proof*⟩

lemma *Sigma_mset_empty1* [*simp*]: $Sigma\_mset\ \{\#\}\ B = \{\#\}$
⟨*proof*⟩

lemma *Sigma_mset_empty2* [*simp*]: $A \times mset\ \{\#\} = \{\#\}$
⟨*proof*⟩

lemma *Sigma_mset_mono*:
assumes $A \subseteq\#\ C$ **and** $\bigwedge x.\ x{\in}\#A \Longrightarrow B\ x \subseteq\#\ D\ x$
shows $Sigma\_mset\ A\ B \subseteq\#\ Sigma\_mset\ C\ D$
⟨*proof*⟩

lemma *mem_Sigma_mset_iff*[*iff*]: $((a,b) \in\#\ Sigma\_mset\ A\ B) = (a \in\#\ A \wedge b \in\#\ B\ a)$
⟨*proof*⟩

lemma *mem_Times_mset_iff*: $x \in\#\ A \times mset\ B \longleftrightarrow fst\ x \in\#\ A \wedge snd\ x \in\#\ B$
⟨*proof*⟩

lemma *Sigma_mset_empty_iff*: $(SIGMAMSET\ i{\in}\#I.\ X\ i) = \{\#\} \longleftrightarrow (\forall i{\in}\#I.\ X\ i = \{\#\})$
⟨*proof*⟩

lemma *Times_mset_subset_mset_cancel1*: $x \in\#\ A \Longrightarrow (A \times mset\ B \subseteq\#\ A \times mset\ C) = (B \subseteq\#\ C)$
⟨*proof*⟩

lemma *Times_mset_subset_mset_cancel2*: $x \in\#\ C \Longrightarrow (A \times mset\ C \subseteq\#\ B \times mset\ C) = (A \subseteq\#\ B)$
⟨*proof*⟩

lemma *Times_mset_eq_cancel2*: $x{\in}\#C \Longrightarrow (A \times mset\ C = B \times mset\ C) = (A = B)$
⟨*proof*⟩

lemma *split_paired_Ball_mset_Sigma_mset* [*simp*, *no_atp*]:
$(\forall z{\in}\#Sigma\_mset\ A\ B.\ P\ z) \longleftrightarrow (\forall x{\in}\#A.\ \forall y{\in}\#B\ x.\ P\ (x,\ y))$
⟨*proof*⟩

lemma *split_paired_Bex_mset_Sigma_mset* [*simp*, *no_atp*]:
$(\exists z \in\# Sigma\_mset\ A\ B.\ P\ z) \longleftrightarrow (\exists x \in\# A.\ \exists y \in\# B\ x.\ P\ (x, y))$
⟨*proof*⟩

lemma *sum_mset_if_eq_constant*:
$(\sum x \in\# M.\ if\ a = x\ then\ (f\ x)\ else\ 0) = ((op\ +\ (f\ a))\ \hat{}\hat{}\ (count\ M\ a))\ 0$
⟨*proof*⟩

lemma *iterate_op_plus*: $((op\ +\ k)\ \hat{}\hat{}\ m)\ 0 = k * m$
⟨*proof*⟩

lemma *untion_image_mset_Pair_distribute*:
$\bigcup\#\{\#image\_mset\ (Pair\ x)\ (C\ x).\ x \in\#\ J - I\#\} = \bigcup\#\{\#image\_mset\ (Pair\ x)\ (C\ x).\ x \in\#\ J\#\} -$
$\bigcup\#\{\#image\_mset\ (Pair\ x)\ (C\ x).\ x \in\#\ I\#\}$
⟨*proof*⟩

lemma *Sigma_mset_Un_distrib1*: $Sigma\_mset\ (I \cup\#\ J)\ C = Sigma\_mset\ I\ C \cup\#\ Sigma\_mset\ J\ C$
⟨*proof*⟩

lemma *Sigma_mset_Un_distrib2*: $(SIGMAMSET\ i \in\# I.\ A\ i \cup\#\ B\ i) = Sigma\_mset\ I\ A \cup\#\ Sigma\_mset\ I\ B$
⟨*proof*⟩

lemma *Sigma_mset_Int_distrib1*: $Sigma\_mset\ (I \cap\#\ J)\ C = Sigma\_mset\ I\ C \cap\#\ Sigma\_mset\ J\ C$
⟨*proof*⟩

lemma *Sigma_mset_Int_distrib2*: $(SIGMAMSET\ i \in\# I.\ A\ i \cap\#\ B\ i) = Sigma\_mset\ I\ A \cap\#\ Sigma\_mset\ I\ B$
⟨*proof*⟩

lemma *Sigma_mset_Diff_distrib1*: $Sigma\_mset\ (I - J)\ C = Sigma\_mset\ I\ C - Sigma\_mset\ J\ C$
⟨*proof*⟩

lemma *Sigma_mset_Diff_distrib2*: $(SIGMAMSET\ i \in\# I.\ A\ i - B\ i) = Sigma\_mset\ I\ A - Sigma\_mset\ I\ B$
⟨*proof*⟩

lemma *Sigma_mset_Union*: $Sigma\_mset\ (\bigcup\#X)\ B = (\bigcup\#\ (image\_mset\ (\lambda A.\ Sigma\_mset\ A\ B)\ X))$
⟨*proof*⟩

lemma *Times_mset_Un_distrib1*: $(A \cup\#\ B) \times mset\ C = A \times mset\ C \cup\#\ B \times mset\ C$
⟨*proof*⟩

lemma *Times_mset_Int_distrib1*: $(A \cap\#\ B) \times mset\ C = A \times mset\ C \cap\#\ B \times mset\ C$
⟨*proof*⟩

lemma *Times_mset_Diff_distrib1*: $(A - B) \times mset\ C = A \times mset\ C - B \times mset\ C$
⟨*proof*⟩

lemma *Times_mset_empty*[*simp*]: $A \times mset\ B = \{\#\} \longleftrightarrow A = \{\#\} \vee B = \{\#\}$
⟨*proof*⟩

lemma *Times_insert_left*: $A \times mset\ add\_mset\ x\ B = A \times mset\ B + image\_mset\ (\lambda a.\ Pair\ a\ x)\ A$
⟨*proof*⟩

lemma *Times_insert_right*: $add\_mset\ a\ A \times mset\ B = A \times mset\ B + image\_mset\ (Pair\ a)\ B$
⟨*proof*⟩

lemma *fst_image_mset_times_mset* [*simp*]:
$image\_mset\ fst\ (A \times mset\ B) = (if\ B = \{\#\}\ then\ \{\#\}\ else\ repeat\_mset\ (size\ B)\ A)$
⟨*proof*⟩

lemma *snd_image_mset_times_mset* [*simp*]:
$image\_mset\ snd\ (A \times mset\ B) = (if\ A = \{\#\}\ then\ \{\#\}\ else\ repeat\_mset\ (size\ A)\ B)$
⟨*proof*⟩

lemma *product_swap_mset*:
image_mset prod.swap (*A* ×*mset B*) = *B* ×*mset A*
⟨*proof*⟩

context
begin

qualified definition *product_mset* :: *'a multiset* ⇒ *'b multiset* ⇒ (*'a* × *'b*) *multiset* **where**
[*code_abbrev*]: *product_mset A B* = *A* ×*mset B*

lemma *member_product_mset*: *x* ∈# *Multiset_More.product_mset A B* ⟷ *x* ∈# *A* ×*mset B*
⟨*proof*⟩

end

lemma *count_Sigma_mset_abs_def*: *count* (*Sigma_mset A B*) = (λ(*a*, *b*) ⇒ *count A a* ∗ *count* (*B a*) *b*)
⟨*proof*⟩

lemma *Times_mset_image_mset1*: *image_mset f A* ×*mset B* = *image_mset* (λ(*a*, *b*). (*f a*, *b*)) (*A* ×*mset B*)
⟨*proof*⟩

lemma *Times_mset_image_mset2*: *A* ×*mset image_mset f B* = *image_mset* (λ(*a*, *b*). (*a*, *f b*)) (*A* ×*mset B*)
⟨*proof*⟩

lemma *sum_le_singleton*: *A* ⊆ {*x*} ⟹ *sum f A* = (*if x* ∈ *A then f x else* 0)
⟨*proof*⟩

lemma *Times_mset_assoc*:
(*A* ×*mset B*) ×*mset C* = *image_mset* (λ(*a*, *b*, *c*). ((*a*, *b*), *c*)) (*A* ×*mset B* ×*mset C*)
⟨*proof*⟩

2.13 Transfer Rules

lemma *plus_multiset_transfer*[*transfer_rule*]:
(*rel_fun* (*rel_mset R*) (*rel_fun* (*rel_mset R*) (*rel_mset R*))) (*op* +) (*op* +)
⟨*proof*⟩

lemma *minus_multiset_transfer*[*transfer_rule*]:
assumes [*transfer_rule*]: *bi_unique R*
shows (*rel_fun* (*rel_mset R*) (*rel_fun* (*rel_mset R*) (*rel_mset R*))) (*op* −) (*op* −)
⟨*proof*⟩

declare *rel_mset_Zero*[*transfer_rule*]

lemma *count_transfer*[*transfer_rule*]:
assumes *bi_unique R*
shows (*rel_fun* (*rel_mset R*) (*rel_fun R op* =)) *count count*
⟨*proof*⟩

lemma *subseteq_multiset_transfer*[*transfer_rule*]:
assumes [*transfer_rule*]: *bi_unique R right_total R*
shows (*rel_fun* (*rel_mset R*) (*rel_fun* (*rel_mset R*) (*op* =)))
(λ*M N*. *filter_mset* (*Domainp R*) *M* ⊆# *filter_mset* (*Domainp R*) *N*) (*op* ⊆#)
⟨*proof*⟩

lemma *sum_mset_transfer*[*transfer_rule*]: *R* 0 0 ⟹ *rel_fun R* (*rel_fun R R*) *op* + *op* + ⟹
(*rel_fun* (*rel_mset R*) *R*) *sum_mset sum_mset*
⟨*proof*⟩

lemma *Sigma_mset_transfer*[*transfer_rule*]:
(*rel_fun* (*rel_mset R*) (*rel_fun* (*rel_fun R* (*rel_mset S*)) (*rel_mset* (*rel_prod R S*))))
Sigma_mset Sigma_mset
⟨*proof*⟩

end

3 Signed (Finite) Multisets

theory *Signed_ Multiset*
imports *Multiset_ More*
abbrevs
$!z = {}_z$
begin

3.1 Definition of Signed Multisets

definition *equiv_ zmset* :: *'a multiset* × *'a multiset* ⇒ *'a multiset* × *'a multiset* ⇒ *bool* **where**
equiv_ zmset = (λ(*Mp*, *Mn*) (*Np*, *Nn*). *Mp* + *Nn* = *Np* + *Mn*)

quotient-type *'a zmultiset* = *'a multiset* × *'a multiset* / *equiv_ zmset*
⟨*proof*⟩

3.2 Basic Operations on Signed Multisets

instantiation *zmultiset* :: (*type*) *cancel_ comm_ monoid_ add*
begin

lift-definition *zero_ zmultiset* :: *'a zmultiset* **is** ({#}, {#}) ⟨*proof*⟩

abbreviation *empty_ zmset* :: *'a zmultiset* ($\{\#\}_z$) **where**
empty_ zmset ≡ *0*

lift-definition *minus_ zmultiset* :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *'a zmultiset* **is**
λ(*Mp*, *Mn*) (*Np*, *Nn*). (*Mp* + *Nn*, *Mn* + *Np*)
⟨*proof*⟩

lift-definition *plus_ zmultiset* :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *'a zmultiset* **is**
λ(*Mp*, *Mn*) (*Np*, *Nn*). (*Mp* + *Np*, *Mn* + *Nn*)
⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation *zmultiset* :: (*type*) *group_ add*
begin

lift-definition *uminus_ zmultiset* :: *'a zmultiset* ⇒ *'a zmultiset* **is** λ(*Mp*, *Mn*). (*Mn*, *Mp*)
⟨*proof*⟩

instance
⟨*proof*⟩

end

lift-definition *zcount* :: *'a zmultiset* ⇒ *'a* ⇒ *int* **is**
λ(*Mp*, *Mn*) *x*. *int* (*count Mp x*) − *int* (*count Mn x*)
⟨*proof*⟩

lemma *zcount_ inject*: *zcount M* = *zcount N* ⟷ *M* = *N*
⟨*proof*⟩

lemma *zmultiset_ eq_ iff*: *M* = *N* ⟷ (∀ *a*. *zcount M a* = *zcount N a*)
⟨*proof*⟩

lemma *zmultiset_eqI*: $(\bigwedge x.\ zcount\ A\ x = zcount\ B\ x) \Longrightarrow A = B$
⟨*proof*⟩

lemma *zcount_uminus*[*simp*]: $zcount\ (-\ A)\ x = -\ zcount\ A\ x$
⟨*proof*⟩

lift-definition *add_zmset* :: $'a \Rightarrow 'a\ zmultiset \Rightarrow 'a\ zmultiset$ **is**
$\lambda x\ (Mp,\ Mn).\ (add\_mset\ x\ Mp,\ Mn)$
⟨*proof*⟩

syntax
_zmultiset :: *args* ⇒ $'a\ zmultiset$ $(\{\#(\_)\#\}_z)$
translations
$\{\#x,\ xs\#\}_z$ == *CONST add_zmset x* $\{\#xs\#\}_z$
$\{\#x\#\}_z$ == *CONST add_zmset x* $\{\#\}_z$

lemma *zcount_empty*[*simp*]: $zcount\ \{\#\}_z\ a = 0$
⟨*proof*⟩

lemma *zcount_add_zmset*[*simp*]:
$zcount\ (add\_zmset\ b\ A)\ a = (if\ b = a\ then\ zcount\ A\ a + 1\ else\ zcount\ A\ a)$
⟨*proof*⟩

lemma *zcount_single*: $zcount\ \{\#b\#\}_z\ a = (if\ b = a\ then\ 1\ else\ 0)$
⟨*proof*⟩

lemma *add_add_same_iff_zmset*[*simp*]: $add\_zmset\ a\ A = add\_zmset\ a\ B \longleftrightarrow A = B$
⟨*proof*⟩

lemma *add_zmset_commute*: $add\_zmset\ x\ (add\_zmset\ y\ M) = add\_zmset\ y\ (add\_zmset\ x\ M)$
⟨*proof*⟩

lemma
singleton_ne_empty_zmset[*simp*]: $\{\#x\#\}_z \neq \{\#\}_z$ **and**
empty_ne_singleton_zmset[*simp*]: $\{\#\}_z \neq \{\#x\#\}_z$
⟨*proof*⟩

lemma
singleton_ne_uminus_singleton_zmset[*simp*]: $\{\#x\#\}_z \neq -\ \{\#y\#\}_z$ **and**
uminus_singleton_ne_singleton_zmset[*simp*]: $-\ \{\#x\#\}_z \neq \{\#y\#\}_z$
⟨*proof*⟩

3.2.1 Conversion to Set and Membership

definition *set_zmset* :: $'a\ zmultiset \Rightarrow 'a\ set$ **where**
$set\_zmset\ M = \{x.\ zcount\ M\ x \neq 0\}$

abbreviation *elem_zmset* :: $'a \Rightarrow 'a\ zmultiset \Rightarrow bool$ **where**
$elem\_zmset\ a\ M \equiv a \in set\_zmset\ M$

notation
$elem\_zmset\ (op\ \in\#_z)$ **and**
$elem\_zmset\ ((\_/\ \in\#_z\ \_)\ [51,\ 51]\ 50)$

notation (*ASCII*)
$elem\_zmset\ (op\ :\#hy)$ **and**
$elem\_zmset\ ((\_/\ :\#hy\ \_)\ [51,\ 51]\ 50)$

abbreviation *not_elem_zmset* :: $'a \Rightarrow 'a\ zmultiset \Rightarrow bool$ **where**
$not\_elem\_zmset\ a\ M \equiv a \notin set\_zmset\ M$

notation
$not\_elem\_zmset\ (op\ \notin\#_z)$ **and**
$not\_elem\_zmset\ ((\_/\ \notin\#_z\ \_)\ [51,\ 51]\ 50)$

notation (*ASCII*)
  not_ elem_ zmset (*op* ~:#*hy*) **and**
  not_ elem_ zmset ((_/ ~:#*hy* _) [*51*, *51*] *50*)

context
begin

qualified abbreviation *Ball* :: *'a zmultiset* ⇒ (*'a* ⇒ *bool*) ⇒ *bool* **where**
  Ball M ≡ *Set.Ball* (*set_ zmset M*)

qualified abbreviation *Bex* :: *'a zmultiset* ⇒ (*'a* ⇒ *bool*) ⇒ *bool* **where**
  Bex M ≡ *Set.Bex* (*set_ zmset M*)

end

syntax
  _*MBall* :: *pttrn* ⇒ *'a set* ⇒ *bool* ⇒ *bool* ((*3*∀ _∈#$_$_./ _) [*0*, *0*, *10*] *10*)
  _*MBex* :: *pttrn* ⇒ *'a set* ⇒ *bool* ⇒ *bool* ((*3*∃ _∈#$_$_./ _) [*0*, *0*, *10*] *10*)

syntax (*ASCII*)
  _*MBall* :: *pttrn* ⇒ *'a set* ⇒ *bool* ⇒ *bool* ((*3*∀ _:#$_$_./ _) [*0*, *0*, *10*] *10*)
  _*MBex* :: *pttrn* ⇒ *'a set* ⇒ *bool* ⇒ *bool* ((*3*∃ _:#$_$_./ _) [*0*, *0*, *10*] *10*)

translations
  ∀ *x*∈#$_z$*A*. *P* ⇌ *CONST Signed_ Multiset.Ball A* (λ*x*. *P*)
  ∃ *x*∈#$_z$*A*. *P* ⇌ *CONST Signed_ Multiset.Bex A* (λ*x*. *P*)

lemma *zcount_ eq_ zero_ iff*: *zcount M x* = *0* ⟷ *x* ∉#$_z$ *M*
  ⟨*proof*⟩

lemma *not_ in_ iff_ zmset*: *x* ∉#$_z$ *M* ⟷ *zcount M x* = *0*
  ⟨*proof*⟩

lemma *zcount_ ne_ zero_ iff*[*simp*]: *zcount M x* ≠ *0* ⟷ *x* ∈#$_z$ *M*
  ⟨*proof*⟩

lemma *zcount_ inI*:
  assumes *zcount M x* = *0* ⟹ *False*
  shows *x* ∈#$_z$ *M*
⟨*proof*⟩

lemma *set_ zmset_ empty*[*simp*]: *set_ zmset* {#}$_z$ = {}
  ⟨*proof*⟩

lemma *set_ zmset_ single*: *set_ zmset* {#*b*#}$_z$ = {*b*}
  ⟨*proof*⟩

lemma *set_ zmset_ eq_ empty_ iff*[*simp*]: *set_ zmset M* = {} ⟷ *M* = {#}$_z$
  ⟨*proof*⟩

lemma *finite_ count_ ne*: *finite* {*x*. *count M x* ≠ *count N x*}
⟨*proof*⟩

lemma *finite_ set_ zmset*[*iff*]: *finite* (*set_ zmset M*)
  ⟨*proof*⟩

lemma *zmultiset_ nonemptyE*[*elim*]:
  assumes *A* ≠ {#}$_z$
  obtains *x* **where** *x* ∈#$_z$ *A*
⟨*proof*⟩

3.2.2 Union

lemma *zcount_ union*[*simp*]: *zcount* (*M* + *N*) *a* = *zcount M a* + *zcount N a*
⟨*proof*⟩

lemma *union_ add_ left_ zmset*[*simp*]: *add_ zmset a A* + *B* = *add_ zmset a* (*A* + *B*)
⟨*proof*⟩

lemma *union_ zmset_ add_ zmset_ right*[*simp*]: *A* + *add_ zmset a B* = *add_ zmset a* (*A* + *B*)
⟨*proof*⟩

lemma *add_ zmset_ add_ single*: ‹*add_ zmset a A* = *A* + {#*a*#}$_z$›
⟨*proof*⟩

3.2.3 Difference

lemma *zcount_ diff*[*simp*]: *zcount* (*M* − *N*) *a* = *zcount M a* − *zcount N a*
⟨*proof*⟩

lemma *add_ zmset_ diff_ bothsides*: ‹*add_ zmset a M* − *add_ zmset a A* = *M* − *A*›
⟨*proof*⟩

lemma *in_ diff_ zcount*: *a* ∈#$_z$ *M* − *N* ⟷ *zcount N a* ≠ *zcount M a*
⟨*proof*⟩

lemma *diff_ add_ zmset*:
fixes *M N Q* :: ′*a zmultiset*
shows *M* − (*N* + *Q*) = *M* − *N* − *Q*
⟨*proof*⟩

lemma *insert_ Diff_ zmset*[*simp*]: *add_ zmset x* (*M* − {#*x*#}$_z$) = *M*
⟨*proof*⟩

lemma *diff_ union_ swap_ zmset*: *add_ zmset b* (*M* − {#*a*#}$_z$) = *add_ zmset b M* − {#*a*#}$_z$
⟨*proof*⟩

lemma *diff_ add_ zmset_ swap*[*simp*]: *add_ zmset b M* − *A* = *add_ zmset b* (*M* − *A*)
⟨*proof*⟩

lemma *diff_ diff_ add_ zmset*[*simp*]: (*M* :: ′*a zmultiset*) − *N* − *P* = *M* − (*N* + *P*)
⟨*proof*⟩

lemma *zmset_ add*[*elim?*]:
obtains *B* **where** *A* = *add_ zmset a B*
⟨*proof*⟩

3.2.4 Equality of Signed Multisets

lemma *single_ eq_ single_ zmset*[*simp*]: {#*a*#}$_z$ = {#*b*#}$_z$ ⟷ *a* = *b*
⟨*proof*⟩

lemma *multi_ self_ add_ other_ not_ self_ zmset*[*simp*]: *M* = *add_ zmset x M* ⟷ *False*
⟨*proof*⟩

lemma *add_ zmset_ remove_ trivial*: ‹*add_ zmset x M* − {#*x*#}$_z$ = *M*›
⟨*proof*⟩

lemma *diff_ single_ eq_ union_ zmset*: *M* − {#*x*#}$_z$ = *N* ⟷ *M* = *add_ zmset x N*
⟨*proof*⟩

lemma *union_ single_ eq_ diff_ zmset*: *add_ zmset x M* = *N* ⟹ *M* = *N* − {#*x*#}$_z$
⟨*proof*⟩

lemma *add_ zmset_ eq_ conv_ diff*:
add_ zmset a M = *add_ zmset b N* ⟷

$M = N \wedge a = b \vee M = add\_zmset\ b\ (N - \{\#a\#\}_z) \wedge N = add\_zmset\ a\ (M - \{\#b\#\}_z)$
⟨*proof*⟩

lemma *add_zmset_eq_conv_ex*:
$(add\_zmset\ a\ M = add\_zmset\ b\ N) =$
$(M = N \wedge a = b \vee (\exists K.\ M = add\_zmset\ b\ K \wedge N = add\_zmset\ a\ K))$
⟨*proof*⟩

lemma *multi_member_split*: $\exists A.\ M = add\_zmset\ x\ A$
⟨*proof*⟩

3.3 Conversions from and to Multisets

lift-definition *zmset_of* :: $'a\ multiset \Rightarrow 'a\ zmultiset$ **is** $\lambda f.\ (Abs\_multiset\ f, \{\#\})$ ⟨*proof*⟩

lemma *zmset_of_inject*[*simp*]: $zmset\_of\ M = zmset\_of\ N \longleftrightarrow M = N$
⟨*proof*⟩

lemma *zmset_of_empty*[*simp*]: $zmset\_of\ \{\#\} = \{\#\}_z$
⟨*proof*⟩

lemma *zmset_of_add_mset*[*simp*]: $zmset\_of\ (add\_mset\ x\ M) = add\_zmset\ x\ (zmset\_of\ M)$
⟨*proof*⟩

lemma *zcount_of_mset*[*simp*]: $zcount\ (zmset\_of\ M)\ x = int\ (count\ M\ x)$
⟨*proof*⟩

lemma *zmset_of_plus*: $zmset\_of\ (M + N) = zmset\_of\ M + zmset\_of\ N$
⟨*proof*⟩

lift-definition *mset_pos* :: $'a\ zmultiset \Rightarrow 'a\ multiset$ **is** $\lambda(Mp, Mn).\ count\ (Mp - Mn)$
⟨*proof*⟩

lift-definition *mset_neg* :: $'a\ zmultiset \Rightarrow 'a\ multiset$ **is** $\lambda(Mp, Mn).\ count\ (Mn - Mp)$
⟨*proof*⟩

lemma
zmset_of_inverse[*simp*]: $mset\_pos\ (zmset\_of\ M) = M$ **and**
minus_zmset_of_inverse[*simp*]: $mset\_neg\ (-\ zmset\_of\ M) = M$
⟨*proof*⟩

lemma *neg_zmset_pos*[*simp*]: $mset\_neg\ (zmset\_of\ M) = \{\#\}$
⟨*proof*⟩

lemma
count_mset_pos[*simp*]: $count\ (mset\_pos\ M)\ x = nat\ (zcount\ M\ x)$ **and**
count_mset_neg[*simp*]: $count\ (mset\_neg\ M)\ x = nat\ (-\ zcount\ M\ x)$
⟨*proof*⟩

lemma
mset_pos_empty[*simp*]: $mset\_pos\ \{\#\}_z = \{\#\}$ **and**
mset_neg_empty[*simp*]: $mset\_neg\ \{\#\}_z = \{\#\}$
⟨*proof*⟩

lemma
mset_pos_singleton[*simp*]: $mset\_pos\ \{\#x\#\}_z = \{\#x\#\}$ **and**
mset_neg_singleton[*simp*]: $mset\_neg\ \{\#x\#\}_z = \{\#\}$
⟨*proof*⟩

lemma
mset_pos_neg_partition: $M = zmset\_of\ (mset\_pos\ M) - zmset\_of\ (mset\_neg\ M)$ **and**
mset_pos_as_neg: $zmset\_of\ (mset\_pos\ M) = zmset\_of\ (mset\_neg\ M) + M$ **and**
mset_neg_as_pos: $zmset\_of\ (mset\_neg\ M) = zmset\_of\ (mset\_pos\ M) - M$
⟨*proof*⟩

lemma *mset_pos_uminus*[*simp*]: *mset_pos* (− *A*) = *mset_neg A*
⟨*proof*⟩

lemma *mset_neg_uminus*[*simp*]: *mset_neg* (− *A*) = *mset_pos A*
⟨*proof*⟩

lemma *mset_pos_plus*[*simp*]:
mset_pos (*A* + *B*) = (*mset_pos A* − *mset_neg B*) + (*mset_pos B* − *mset_neg A*)
⟨*proof*⟩

lemma *mset_neg_plus*[*simp*]:
mset_neg (*A* + *B*) = (*mset_neg A* − *mset_pos B*) + (*mset_neg B* − *mset_pos A*)
⟨*proof*⟩

lemma *mset_pos_diff*[*simp*]:
mset_pos (*A* − *B*) = (*mset_pos A* − *mset_pos B*) + (*mset_neg B* − *mset_neg A*)
⟨*proof*⟩

lemma *mset_neg_diff*[*simp*]:
mset_neg (*A* − *B*) = (*mset_neg A* − *mset_neg B*) + (*mset_pos B* − *mset_pos A*)
⟨*proof*⟩

lemma *mset_pos_neg_dual*:
mset_pos a + *mset_pos b* + (*mset_neg a* − *mset_pos b*) + (*mset_neg b* − *mset_pos a*) =
mset_neg a + *mset_neg b* + (*mset_pos a* − *mset_neg b*) + (*mset_pos b* − *mset_neg a*)
⟨*proof*⟩

lemma *decompose_zmset_of2*:
obtains *A B C* **where**
M = *zmset_of A* + *C* **and**
N = *zmset_of B* + *C*
⟨*proof*⟩

3.3.1 Pointwise Ordering Induced by *zcount*

definition *subseteq_zmset* :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *bool* (**infix** ⊆#$_z$ *50*) **where**
A ⊆#$_z$ *B* ⟷ (∀ *a*. *zcount A a* ≤ *zcount B a*)

definition *subset_zmset* :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *bool* (**infix** ⊂#$_z$ *50*) **where**
A ⊂#$_z$ *B* ⟷ *A* ⊆#$_z$ *B* ∧ *A* ≠ *B*

abbreviation (*input*)
supseteq_zmset :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *bool* (**infix** ⊇#$_z$ *50*)
where
supseteq_zmset A B ≡ *B* ⊆#$_z$ *A*

abbreviation (*input*)
supset_zmset :: *'a zmultiset* ⇒ *'a zmultiset* ⇒ *bool* (**infix** ⊃#$_z$ *50*)
where
supset_zmset A B ≡ *B* ⊂#$_z$ *A*

notation (*input*)
subseteq_zmset (**infix** ≤#$_z$ *50*) **and**
supseteq_zmset (**infix** ≥#$_z$ *50*)

notation (*ASCII*)
subseteq_zmset (**infix** <=#$_z$ *50*) **and**
subset_zmset (**infix** <#$_z$ *50*) **and**
supseteq_zmset (**infix** >=#$_z$ *50*) **and**
supset_zmset (**infix** >#$_z$ *50*)

interpretation *subset_zmset*: *ordered_ab_semigroup_add_imp_le op* + *op* − *op* ⊆#$_z$ *op* ⊂#$_z$
⟨*proof*⟩

interpretation *subset_ zmset*:
ordered_ ab_ semigroup_ monoid_ add_ imp_ le op + 0 op − op $\leq\#_z$ *op* $<\#_z$
⟨*proof*⟩

lemma *zmset_ subset_ eqI*: $(\bigwedge a.\ zcount\ A\ a \leq zcount\ B\ a) \Longrightarrow A \subseteq\#_z B$
⟨*proof*⟩

lemma *zmset_ subset_ eq_ zcount*: $A \subseteq\#_z B \Longrightarrow zcount\ A\ a \leq zcount\ B\ a$
⟨*proof*⟩

lemma *zmset_ subset_ eq_ add_ zmset_ cancel*: ‹*add_ zmset a A* $\subseteq\#_z$ *add_ zmset a B* $\longleftrightarrow$ *A* $\subseteq\#_z$ *B*›
⟨*proof*⟩

lemma *zmset_ subset_ eq_ zmultiset_ union_ diff_ commute*:
$A - B + C = A + C - B$ **for** *A B C* :: *'a zmultiset*
⟨*proof*⟩

lemma *zmset_ subset_ eq_ insertD*: *add_ zmset x A* $\subseteq\#_z$ *B* $\Longrightarrow$ *A* $\subset\#_z$ *B*
⟨*proof*⟩

lemma *zmset_ subset_ insertD*: *add_ zmset x A* $\subset\#_z$ *B* $\Longrightarrow$ *A* $\subset\#_z$ *B*
⟨*proof*⟩

lemma *subset_ eq_ diff_ conv_ zmset*: *A − C* $\subseteq\#_z$ *B* $\longleftrightarrow$ *A* $\subseteq\#_z$ *B + C*
⟨*proof*⟩

lemma *multi_ psub_ of_ add_ self_ zmset*[*simp*]: *A* $\subset\#_z$ *add_ zmset x A*
⟨*proof*⟩

lemma *multi_ psub_ self_ zmset*: *A* $\subset\#_z$ *A = False*
⟨*proof*⟩

lemma *zmset_ subset_ add_ zmset*[*simp*]: *add_ zmset x N* $\subset\#_z$ *add_ zmset x M* $\longleftrightarrow$ *N* $\subset\#_z$ *M*
⟨*proof*⟩

lemma *zmset_ of_ subseteq_ iff*[*simp*]: *zmset_ of M* $\subseteq\#_z$ *zmset_ of N* $\longleftrightarrow$ *M* $\subseteq\#$ *N*
⟨*proof*⟩

lemma *zmset_ of_ subset_ iff*[*simp*]: *zmset_ of M* $\subset\#_z$ *zmset_ of N* $\longleftrightarrow$ *M* $\subset\#$ *N*
⟨*proof*⟩

lemma
mset_ pos_ supset: *A* $\subseteq\#_z$ *zmset_ of* (*mset_ pos A*) **and**
mset_ neg_ supset: *− A* $\subseteq\#_z$ *zmset_ of* (*mset_ neg A*)
⟨*proof*⟩

lemma *subset_ mset_ zmsetE*:
assumes *M* $\subset\#_z$ *N*
obtains *A B C* **where**
M = zmset_ of A + C **and** *N = zmset_ of B + C* **and** *A* $\subset\#$ *B*
⟨*proof*⟩

lemma *subseteq_ mset_ zmsetE*:
assumes *M* $\subseteq\#_z$ *N*
obtains *A B C* **where**
M = zmset_ of A + C **and** *N = zmset_ of B + C* **and** *A* $\subseteq\#$ *B*
⟨*proof*⟩

3.3.2 Subset is an Order

interpretation *subset_ zmset*: *order op* $\subseteq\#_z$ *op* $\subset\#_z$
⟨*proof*⟩

3.4 Replicate and Repeat Operations

definition *replicate_zmset* :: *nat* ⇒ *'a* ⇒ *'a zmultiset* **where**
replicate_zmset n x = (*add_zmset x* ^^ *n*) {#}$_z$

lemma *replicate_zmset_0*[*simp*]: *replicate_zmset 0 x* = {#}$_z$
⟨*proof*⟩

lemma *replicate_zmset_Suc*[*simp*]: *replicate_zmset* (*Suc n*) *x* = *add_zmset x* (*replicate_zmset n x*)
⟨*proof*⟩

lemma *count_replicate_zmset*[*simp*]:
zcount (*replicate_zmset n x*) *y* = (*if y* = *x then of_nat n else 0*)
⟨*proof*⟩

fun *repeat_zmset* :: *nat* ⇒ *'a zmultiset* ⇒ *'a zmultiset* **where**
repeat_zmset 0 _ = {#}$_z$ |
repeat_zmset (*Suc n*) *A* = *A* + *repeat_zmset n A*

lemma *count_repeat_zmset*[*simp*]: *zcount* (*repeat_zmset i A*) *a* = *of_nat i* ∗ *zcount A a*
⟨*proof*⟩

lemma *repeat_zmset_right*[*simp*]: *repeat_zmset a* (*repeat_zmset b A*) = *repeat_zmset* (*a* ∗ *b*) *A*
⟨*proof*⟩

lemma *left_diff_repeat_zmset_distrib'*:
‹*i* ≥ *j* ⟹ *repeat_zmset* (*i* − *j*) *u* = *repeat_zmset i u* − *repeat_zmset j u*›
⟨*proof*⟩

lemma *left_add_mult_distrib_zmset*:
repeat_zmset i u + (*repeat_zmset j u* + *k*) = *repeat_zmset* (*i*+*j*) *u* + *k*
⟨*proof*⟩

lemma *repeat_zmset_distrib*: *repeat_zmset* (*m* + *n*) *A* = *repeat_zmset m A* + *repeat_zmset n A*
⟨*proof*⟩

lemma *repeat_zmset_distrib2*[*simp*]:
repeat_zmset n (*A* + *B*) = *repeat_zmset n A* + *repeat_zmset n B*
⟨*proof*⟩

lemma *repeat_zmset_replicate_zmset*[*simp*]: *repeat_zmset n* {#*a*#}$_z$ = *replicate_zmset n a*
⟨*proof*⟩

lemma *repeat_zmset_distrib_add_zmset*[*simp*]:
repeat_zmset n (*add_zmset a A*) = *replicate_zmset n a* + *repeat_zmset n A*
⟨*proof*⟩

lemma *repeat_zmset_empty*[*simp*]: *repeat_zmset n* {#}$_z$ = {#}$_z$
⟨*proof*⟩

3.4.1 Filter (with Comprehension Syntax)

lift-definition *filter_zmset* :: (*'a* ⇒ *bool*) ⇒ *'a zmultiset* ⇒ *'a zmultiset* **is**
λ*P* (*Mp*, *Mn*). (*filter_mset P Mp*, *filter_mset P Mn*)
⟨*proof*⟩

syntax (*ASCII*)
*MCollect* :: *pttrn* ⇒ *'a zmultiset* ⇒ *bool* ⇒ *'a zmultiset* ((1{# :#*hy* _./ _#}))
syntax
*MCollect* :: *pttrn* ⇒ *'a zmultiset* ⇒ *bool* ⇒ *'a zmultiset* ((1{# ∈#$_z$ _./ _#}))
translations
{#*x* ∈#$_z$ *M*. *P*#} == *CONST filter_zmset* (λ*x*. *P*) *M*

lemma *count_filter_zmset*[*simp*]:

zcount (*filter_ zmset P M*) *a* = (*if P a then zcount M a else 0*)
⟨*proof*⟩

lemma *filter_ empty_ zmset*[*simp*]: *filter_ zmset P* {#}$_z$ = {#}$_z$
⟨*proof*⟩

lemma *filter_ single_ zmset*: *filter_ zmset P* {#*x*#}$_z$ = (*if P x then* {#*x*#}$_z$ *else* {#}$_z$)
⟨*proof*⟩

lemma *filter_ union_ zmset*[*simp*]: *filter_ zmset P* (*M* + *N*) = *filter_ zmset P M* + *filter_ zmset P N*
⟨*proof*⟩

lemma *filter_ diff_ zmset*[*simp*]: *filter_ zmset P* (*M* − *N*) = *filter_ zmset P M* − *filter_ zmset P N*
⟨*proof*⟩

lemma *filter_ add_ zmset*[*simp*]:
filter_ zmset P (*add_ zmset x A*) =
(*if P x then add_ zmset x* (*filter_ zmset P A*) *else filter_ zmset P A*)
⟨*proof*⟩

lemma *zmultiset_ filter_ mono*:
assumes *A* ⊆#$_z$ *B*
shows *filter_ zmset f A* ⊆#$_z$ *filter_ zmset f B*
⟨*proof*⟩

lemma *filter_ filter_ zmset*: *filter_ zmset P* (*filter_ zmset Q M*) = {#*x* ∈# *M*. *Q x* ∧ *P x*#}
⟨*proof*⟩

lemma
filter_ zmset_ True[*simp*]: {#*y* ∈#$_z$ *M*. *True*#} = *M* **and**
filter_ zmset_ False[*simp*]: {#*y* ∈#$_z$ *M*. *False*#} = {#}$_z$
⟨*proof*⟩

3.5 Uncategorized

lemma *multi_ drop_ mem_ not_ eq_ zmset*: *B* − {#*c*#}$_z$ ≠ *B*
⟨*proof*⟩

lemma *zmultiset_ partition*: *M* = {#*x* ∈#$_z$ *M*. *P x* #} + {#*x* ∈#$_z$ *M*. ¬ *P x*#}
⟨*proof*⟩

3.6 Image

definition *image_ zmset* :: ('*a* ⇒ '*b*) ⇒ '*a zmultiset* ⇒ '*b zmultiset* **where**
image_ zmset f M =
zmset_ of (*fold_ mset* (*add_ mset* ∘ *f*) {#} (*mset_ pos M*)) −
zmset_ of (*fold_ mset* (*add_ mset* ∘ *f*) {#} (*mset_ neg M*))

3.7 Multiset Order

instantiation *zmultiset* :: (*preorder*) *order*
begin

lift-definition *less_ zmultiset* :: '*a zmultiset* ⇒ '*a zmultiset* ⇒ *bool* **is**
λ(*Mp*, *Mn*) (*Np*, *Nn*). *Mp* + *Nn* < *Mn* + *Np*
⟨*proof*⟩

definition *less_ eq_ zmultiset* :: '*a zmultiset* ⇒ '*a zmultiset* ⇒ *bool* **where**
less_ eq_ zmultiset M' M ⟷ *M'* < *M* ∨ *M'* = *M*

instance
⟨*proof*⟩

end

instance *zmultiset* :: (*preorder*) *ordered_cancel_comm_monoid_add*
⟨*proof*⟩

instance *zmultiset* :: (*preorder*) *ordered_ab_group_add*
⟨*proof*⟩

instantiation *zmultiset* :: (*linorder*) *distrib_lattice*
begin

definition *inf_zmultiset* :: 'a *zmultiset* ⇒ 'a *zmultiset* ⇒ 'a *zmultiset* **where**
inf_zmultiset A B = (*if A* < *B then A else B*)

definition *sup_zmultiset* :: 'a *zmultiset* ⇒ 'a *zmultiset* ⇒ 'a *zmultiset* **where**
sup_zmultiset A B = (*if B* > *A then B else A*)

lemma *not_lt_iff_ge_zmset*: ¬ *x* < *y* ⟷ *x* ≥ *y* **for** *x y* :: 'a *zmultiset*
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *zmset_of_less*: *zmset_of M* < *zmset_of N* ⟷ *M* < *N*
⟨*proof*⟩

lemma *zmset_of_le*: *zmset_of M* ≤ *zmset_of N* ⟷ *M* ≤ *N*
⟨*proof*⟩

instance *zmultiset* :: (*preorder*) *ordered_ab_semigroup_add*
⟨*proof*⟩

instance *zmultiset* :: (*preorder*) *ordered_ab_semigroup_add_imp_le*
⟨*proof*⟩

instance *zmultiset* :: (*linorder*) *linordered_cancel_ab_semigroup_add*
⟨*proof*⟩

lemma *less_mset_zmsetE*:
assumes *M* < *N*
obtains *A B C* **where**
M = *zmset_of A* + *C* **and** *N* = *zmset_of B* + *C* **and** *A* < *B*
⟨*proof*⟩

lemma *less_eq_mset_zmsetE*:
assumes *M* ≤ *N*
obtains *A B C* **where**
M = *zmset_of A* + *C* **and** *N* = *zmset_of B* + *C* **and** *A* ≤ *B*
⟨*proof*⟩

lemma *subset_eq_imp_le_zmset*: *M* ⊆#$_z$ *N* ⟹ *M* ≤ *N*
⟨*proof*⟩

lemma *subset_imp_less_zmset*: *M* ⊂#$_z$ *N* ⟹ *M* < *N*
⟨*proof*⟩

lemma *lt_imp_ex_zcount_lt*:
assumes *m_lt_n*: *M* < *N*
shows ∃ *y*. *zcount M y* < *zcount N y*
⟨*proof*⟩

end

4 Nested Multisets

theory *Nested_Multiset*
imports ~~/src/HOL/Library/Multiset_Order
begin

declare *multiset.map_comp* [*simp*]
declare *multiset.map_cong* [*cong*]

4.1 Type Definition

datatype *'a nmultiset* =
  Elem 'a
| *MSet 'a nmultiset multiset*

inductive *no_elem* **where**
  $(\bigwedge X.\ X \in\# M \Longrightarrow$ *no_elem* $X) \Longrightarrow$ *no_elem* (*MSet* M)

inductive-set *nmultiset_sub* **where**
  $X \in\# M \Longrightarrow (X,$ *MSet* $M) \in$ *nmultiset_sub*

lemma *wf_nmultiset_sub*[*simp*]: *wf nmultiset_sub*
⟨*proof*⟩

primrec *depth_nmultiset* (|_|) **where**
  |*Elem a*| = *0*
| |*MSet M*| =
    (*let X* = *set_mset* (*image_mset depth_nmultiset M*)
    in if X = {} *then 0 else Suc* (*Max X*))

lemma *depth_nmultiset_MSet*: $x \in\# M \Longrightarrow |x| <$ |*MSet* M|
  ⟨*proof*⟩

declare *depth_nmultiset.simps*(*2*)[*simp del*]

4.2 Dershowitz and Manna's Nested Multiset Order

The Dershowitz–Manna extension:

definition *less_multiset_ext*$_{DM}$ **where**
  less_multiset_ext$_{DM}$ $R\ M\ N \longleftrightarrow$
   $(\exists X\ Y.\ X \neq \{\#\} \wedge X \leq\# N \wedge M = (N - X) + Y \wedge (\forall k.\ k \in\# Y \longrightarrow (\exists a.\ a \in\# X \wedge R\ k\ a)))$

lemma *less_multiset_ext*$_{DM}$*_imp_mult*:
  assumes
    N_A: *set_mset N* $\subseteq A$ **and** *M_A*: *set_mset M* $\subseteq A$ **and** *less*: *less_multiset_ext*$_{DM}$ $R\ M\ N$
  shows $(M, N) \in$ *mult* $\{(x, y).\ x \in A \wedge y \in A \wedge R\ x\ y\}$
⟨*proof*⟩

lemma *mult_imp_less_multiset_ext*$_{DM}$:
  assumes
    N_A: *set_mset N* $\subseteq A$ **and** *M_A*: *set_mset M* $\subseteq A$ **and**
    trans: $\forall x \in A.\ \forall y \in A.\ \forall z \in A.\ R\ x\ y \longrightarrow R\ y\ z \longrightarrow R\ x\ z$ **and**
    in_mult: $(M, N) \in$ *mult* $\{(x, y).\ x \in A \wedge y \in A \wedge R\ x\ y\}$
  shows *less_multiset_ext*$_{DM}$ $R\ M\ N$
  ⟨*proof*⟩

lemma *less_multiset_ext*$_{DM}$*_iff_mult*:
  assumes
    N_A: *set_mset N* $\subseteq A$ **and** *M_A*: *set_mset M* $\subseteq A$ **and**
    trans: $\forall x \in A.\ \forall y \in A.\ \forall z \in A.\ R\ x\ y \longrightarrow R\ y\ z \longrightarrow R\ x\ z$
  shows *less_multiset_ext*$_{DM}$ $R\ M\ N \longleftrightarrow (M, N) \in$ *mult* $\{(x, y).\ x \in A \wedge y \in A \wedge R\ x\ y\}$
  ⟨*proof*⟩

instantiation *nmultiset* :: (*preorder*) *preorder*
begin

lemma *less_multiset_ext$_{DM}$_cong*[*fundef_cong*]:
(⋀*X Y k a*. *X* ≠ {#} ⟹ *X* ≤# *N* ⟹ *M* = (*N* − *X*) + *Y* ⟹ *k* ∈# *Y* ⟹ *R k a* = *S k a*) ⟹
less_multiset_ext$_{DM}$ R M N = *less_multiset_ext$_{DM}$ S M N*
⟨*proof*⟩

function (*sequential*) *less_nmultiset* :: 'a *nmultiset* ⇒ 'a *nmultiset* ⇒ *bool* **where**
less_nmultiset (*Elem a*) (*Elem b*) = (*a* < *b*)
| *less_nmultiset* (*Elem a*) (*MSet M*) = *True*
| *less_nmultiset* (*MSet M*) (*Elem a*) = *False*
| *less_nmultiset* (*MSet M*) (*MSet N*) = *less_multiset_ext$_{DM}$ less_nmultiset M N*
⟨*proof*⟩
termination
⟨*proof*⟩

lemmas *less_nmultiset_induct* =
less_nmultiset.induct[*case_names Elem_Elem Elem_MSet MSet_Elem MSet_MSet*]

lemmas *less_nmultiset_cases* =
less_nmultiset.cases[*case_names Elem_Elem Elem_MSet MSet_Elem MSet_MSet*]

lemma *trans_less_nmultiset*:
fixes *X Y Z* :: 'a *nmultiset*
shows *X* < *Y* ⟹ *Y* < *Z* ⟹ *X* < *Z*
⟨*proof*⟩

lemma *irrefl_less_nmultiset*:
fixes *X* :: 'a *nmultiset*
shows *X* < *X* ⟹ *False*
⟨*proof*⟩

lemma *antisym_less_nmultiset*:
fixes *X Y* :: 'a *nmultiset*
shows *X* < *Y* ⟹ *Y* < *X* ⟹ *False*
⟨*proof*⟩

definition *less_eq_nmultiset* :: 'a *nmultiset* ⇒ 'a *nmultiset* ⇒ *bool* **where**
less_eq_nmultiset X Y = (*X* < *Y* ∨ *X* = *Y*)

instance
⟨*proof*⟩

lemma *less_multiset_ext$_{DM}$_less*: *less_multiset_ext$_{DM}$ op* < = (*op* <)
⟨*proof*⟩

end

instantiation *nmultiset* :: (*order*) *order*
begin

instance
⟨*proof*⟩

end

instantiation *nmultiset* :: (*linorder*) *linorder*
begin

lemma *total_less_nmultiset*:
fixes *X Y* :: 'a *nmultiset*
shows ¬ *X* < *Y* ⟹ *Y* ≠ *X* ⟹ *Y* < *X*

⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *less_depth_nmultiset_imp_less_nmultiset*:
$|X| < |Y| \Longrightarrow X < Y$
⟨*proof*⟩

lemma *less_nmultiset_imp_le_depth_nmultiset*:
$X < Y \Longrightarrow |X| \leq |Y|$
⟨*proof*⟩

lemma *eq_mlex_I*:
fixes $f :: 'a \Rightarrow nat$ **and** $R :: 'a \Rightarrow 'a \Rightarrow bool$
assumes $\bigwedge X\ Y.\ f\ X < f\ Y \Longrightarrow R\ X\ Y$ **and** *antisymP R*
shows $\{(X,\ Y).\ R\ X\ Y\} = f\ <*mlex*>\ \{(X,\ Y).\ f\ X = f\ Y \wedge R\ X\ Y\}$
⟨*proof*⟩

instantiation *nmultiset* :: (*wellorder*) *wellorder*
begin

lemma *depth_nmultiset_eq_0*[*simp*]: $|X| = 0 \longleftrightarrow (X = MSet\ \{\#\} \vee (\exists x.\ X = Elem\ x))$
⟨*proof*⟩

lemma *depth_nmultiset_eq_Suc*[*simp*]: $|X| = Suc\ n \longleftrightarrow$
$(\exists N.\ X = MSet\ N \wedge (\exists Y \in\# N.\ |Y| = n) \wedge (\forall Y \in\# N.\ |Y| \leq n))$
⟨*proof*⟩

lemma *wf_less_nmultiset_depth*:
wf $\{(X :: 'a\ nmultiset,\ Y).\ |X| = i \wedge |Y| = i \wedge X < Y\}$
⟨*proof*⟩

lemma *wf_less_nmultiset*: *wf* $\{(X :: 'a\ nmultiset,\ Y :: 'a\ nmultiset).\ X < Y\}$ (**is** *wf ?R*)
⟨*proof*⟩

instance ⟨*proof*⟩

end

end

5 Hereditar(il)y (Finite) Multisets

theory *Hereditary_Multiset*
imports *Multiset_More Nested_Multiset*
begin

5.1 Type Definition

datatype *hmultiset* =
HMSet (*hmsetmset*: *hmultiset multiset*)

lemma *hmsetmset_inject*[*simp*]: *hmsetmset A* = *hmsetmset B* $\longleftrightarrow$ *A* = *B*
⟨*proof*⟩

primrec *Rep_hmultiset* :: *hmultiset* ⇒ *unit nmultiset* **where**
Rep_hmultiset (*HMSet M*) = *MSet* (*image_mset Rep_hmultiset M*)

primrec (*nonexhaustive*) *Abs_hmultiset* :: *unit nmultiset* ⇒ *hmultiset* **where**
Abs_hmultiset (*MSet M*) = *HMSet* (*image_mset Abs_hmultiset M*)

lemma *type_definition_hmultiset*:
type_definition Rep_hmultiset Abs_hmultiset {*X*. *no_elem X*}
⟨*proof*⟩

setup-lifting *type_definition_hmultiset*

lemma *HMSet_alt*: *HMSet* = *Abs_hmultiset o MSet o image_mset Rep_hmultiset*
⟨*proof*⟩

lemma *HMSet_transfer*[*transfer_rule*]: *rel_fun* (*rel_mset pcr_hmultiset*) *pcr_hmultiset MSet HMSet*
⟨*proof*⟩

5.2 Restriction of Dershowitz and Manna's Nested Multiset Order

instantiation *hmultiset* :: *linorder*
begin

lift-definition *less_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *bool* **is** *op* < ⟨*proof*⟩
lift-definition *less_eq_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *bool* **is** *op* ≤ ⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *less_HMSet_iff_less_multiset_ext*$_{DM}$: *HMSet M* < *HMSet N* ⟷ *less_multiset_ext*$_{DM}$ (*op* <) *M N*
⟨*proof*⟩

lemma *hmsetmset_less*[*simp*]: *hmsetmset M* < *hmsetmset N* ⟷ *M* < *N*
⟨*proof*⟩

lemma *hmsetmset_le*[*simp*]: *hmsetmset M* ≤ *hmsetmset N* ⟷ *M* ≤ *N*
⟨*proof*⟩

lemma *wf_less_hmultiset*: *wf* {(*X* :: *hmultiset*, *Y* :: *hmultiset*). *X* < *Y*}
⟨*proof*⟩

instance *hmultiset* :: *wellorder*
⟨*proof*⟩

lemma *HMSet_less*[*simp*]: *HMSet M* < *HMSet N* ⟷ *M* < *N*
⟨*proof*⟩

lemma *HMSet_le*[*simp*]: *HMSet M* ≤ *HMSet N* ⟷ *M* ≤ *N*
⟨*proof*⟩

inductive-set *hmultiset_sub* **where**
X ∈# *M* ⟹ (*X*, *HMSet M*) ∈ *hmultiset_sub*

lemma *wf_hmultiset_sub*[*simp*]: *wf hmultiset_sub*
⟨*proof*⟩

5.3 Disjoint Union and Truncated Difference

instantiation *hmultiset* :: *cancel_comm_monoid_add*
begin

definition *zero_hmultiset* :: *hmultiset* **where**
0 = *HMSet* {#}

lemma *hmsetmset_empty_iff*[*simp*]: *hmsetmset n* = {#} ⟷ *n* = *0*
⟨*proof*⟩

lemma
HMSet_eq_0_iff[*simp*]: *HMSet m* = *0* ⟷ *m* = {#} **and**
zero_eq_HMSet[*simp*]: *0* = *HMSet m* ⟷ *m* = {#}
⟨*proof*⟩

definition *plus_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *hmultiset* **where**
A + *B* = *HMSet* (*hmsetmset A* + *hmsetmset B*)

definition *minus_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *hmultiset* **where**
A − *B* = *HMSet* (*hmsetmset A* − *hmsetmset B*)

instance
⟨*proof*⟩

end

lemma *HMSet_plus*: *HMSet* (*A* + *B*) = *HMSet A* + *HMSet B*
⟨*proof*⟩

lemma *HMSet_diff*: *HMSet* (*A* − *B*) = *HMSet A* − *HMSet B*
⟨*proof*⟩

lemma *hmsetmset_plus*: *hmsetmset* (*M* + *N*) = *hmsetmset M* + *hmsetmset N*
⟨*proof*⟩

lemma *hmsetmset_diff*: *hmsetmset* (*M* − *N*) = *hmsetmset M* − *hmsetmset N*
⟨*proof*⟩

lemma *diff_diff_add_hmset*[*simp*]: *a* − *b* − *c* = *a* − (*b* + *c*) **for** *a b c* :: *hmultiset*
⟨*proof*⟩

instance *hmultiset* :: *comm_monoid_diff*
⟨*proof*⟩

instantiation *hmultiset* :: *order_bot*
begin

definition *bot_hmultiset* :: *hmultiset* **where**
bot_hmultiset = *0*

instance
⟨*proof*⟩

end

instance *hmultiset* :: *no_top*
⟨*proof*⟩

instance *hmultiset* :: *ordered_cancel_comm_monoid_add*
⟨*proof*⟩

instance *hmultiset* :: *ordered_ab_semigroup_add_imp_le*
⟨*proof*⟩

lemma *le_minus_plus_same_hmset*: *m* ≤ *m* − *n* + *n* **for** *m n* :: *hmultiset*
⟨*proof*⟩

5.4 Infimum and Supremum

instantiation *hmultiset* :: *distrib_lattice*
begin

definition *inf_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *hmultiset* **where**
inf_hmultiset A B = (*if A* < *B then A else B*)

definition *sup_ hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *hmultiset* **where**
sup_ hmultiset A B = (*if B* > *A then B else A*)

instance
⟨*proof*⟩

end

end

6 Signed Hereditar(il)y (Finite) Multisets

theory *Signed_ Hereditary_ Multiset*
imports *Signed_ Multiset Hereditary_ Multiset*
begin

6.1 Type Definition

typedef *zhmultiset* = *UNIV* :: *hmultiset zmultiset set*
morphisms *zhmsetmset ZHMSet*
⟨*proof*⟩

lemmas *ZHMSet_ inverse*[*simp*] = *ZHMSet_ inverse*[*OF UNIV_ I*]
lemmas *ZHMSet_ inject*[*simp*] = *ZHMSet_ inject*[*OF UNIV_ I UNIV_ I*]

declare
zhmsetmset_ inverse [*simp*]
zhmsetmset_ inject [*simp*]

setup-lifting *type_ definition_ zhmultiset*

6.2 Multiset Order

instantiation *zhmultiset* :: *linorder*
begin

lift-definition *less_ zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *bool* **is** *op* < ⟨*proof*⟩
lift-definition *less_ eq_ zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *bool* **is** *op* ≤ ⟨*proof*⟩

instance
⟨*proof*⟩

end

lemmas *ZHMSet_ less*[*simp*] = *less_ zhmultiset.abs_ eq*
lemmas *ZHMSet_ le*[*simp*] = *less_ eq_ zhmultiset.abs_ eq*
lemmas *zhmsetmset_ less*[*simp*] = *less_ zhmultiset.rep_ eq*[*symmetric*]
lemmas *zhmsetmset_ le*[*simp*] = *less_ eq_ zhmultiset.rep_ eq*[*symmetric*]

6.3 Embedding and Projections of Syntactic Ordinals

abbreviation *zhmset_ of* :: *hmultiset* ⇒ *zhmultiset* **where**
zhmset_ of M ≡ *ZHMSet* (*zmset_ of* (*hmsetmset M*))

lemma *zhmset_ of_ inject*[*simp*]: *zhmset_ of M* = *zhmset_ of N* ⟷ *M* = *N*
⟨*proof*⟩

lemma *zhmset_ of_ less*: *zhmset_ of M* < *zhmset_ of N* ⟷ *M* < *N*
⟨*proof*⟩

lemma *zhmset_ of_ le*: *zhmset_ of M* ≤ *zhmset_ of N* ⟷ *M* ≤ *N*
⟨*proof*⟩

abbreviation *hmset_pos* :: *zhmultiset* ⇒ *hmultiset* **where**
hmset_pos M ≡ *HMSet* (*mset_pos* (*zhmsetmset M*))

abbreviation *hmset_neg* :: *zhmultiset* ⇒ *hmultiset* **where**
hmset_neg M ≡ *HMSet* (*mset_neg* (*zhmsetmset M*))

6.4 Disjoint Union and Difference

instantiation *zhmultiset* :: *cancel_comm_monoid_add*
begin

lift-definition *zero_zhmultiset* :: *zhmultiset* **is** $\{\#\}_z$ ⟨*proof*⟩

lift-definition *plus_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *zhmultiset* **is**
λ*A B*. *A* + *B* ⟨*proof*⟩

lift-definition *minus_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *zhmultiset* **is**
λ*A B*. *A* − *B* ⟨*proof*⟩

lemmas *ZHMSet_plus* = *plus_zhmultiset.abs_eq*[*symmetric*]
lemmas *ZHMSet_diff* = *minus_zhmultiset.abs_eq*[*symmetric*]
lemmas *zhmsetmset_plus* = *plus_zhmultiset.rep_eq*
lemmas *zhmsetmset_diff* = *minus_zhmultiset.rep_eq*

lemma *zhmset_of_plus*: *zhmset_of* (*A* + *B*) = *zhmset_of A* + *zhmset_of B*
⟨*proof*⟩

lemma *hmsetmset_0*[*simp*]: *hmsetmset 0* = {#}
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *zhmset_of_0*: *zhmset_of 0* = *0*
⟨*proof*⟩

lemma *hmset_pos_plus*:
hmset_pos (*A* + *B*) = (*hmset_pos A* − *hmset_neg B*) + (*hmset_pos B* − *hmset_neg A*)
⟨*proof*⟩

lemma *hmset_neg_plus*:
hmset_neg (*A* + *B*) = (*hmset_neg A* − *hmset_pos B*) + (*hmset_neg B* − *hmset_pos A*)
⟨*proof*⟩

lemma *zhmset_pos_neg_partition*: *M* = *zhmset_of* (*hmset_pos M*) − *zhmset_of* (*hmset_neg M*)
⟨*proof*⟩

lemma *zhmset_pos_as_neg*: *zhmset_of* (*hmset_pos M*) = *zhmset_of* (*hmset_neg M*) + *M*
⟨*proof*⟩

lemma *zhmset_neg_as_pos*: *zhmset_of* (*hmset_neg M*) = *zhmset_of* (*hmset_pos M*) − *M*
⟨*proof*⟩

lemma *hmset_pos_neg_dual*:
hmset_pos a + *hmset_pos b* + (*hmset_neg a* − *hmset_pos b*) + (*hmset_neg b* − *hmset_pos a*) =
hmset_neg a + *hmset_neg b* + (*hmset_pos a* − *hmset_neg b*) + (*hmset_pos b* − *hmset_neg a*)
⟨*proof*⟩

lemma *zhmset_of_sum_list*: *zhmset_of* (*sum_list Ms*) = *sum_list* (*map zhmset_of Ms*)
⟨*proof*⟩

lemma *less_hmset_zhmsetE*:
assumes *m_lt_n*: $M < N$
obtains $A\ B\ C$ **where** $M = zhmset\_of\ A + C$ **and** $N = zhmset\_of\ B + C$ **and** $A < B$
⟨*proof*⟩

lemma *less_eq_hmset_zhmsetE*:
assumes *m_le_n*: $M \leq N$
obtains $A\ B\ C$ **where** $M = zhmset\_of\ A + C$ **and** $N = zhmset\_of\ B + C$ **and** $A \leq B$
⟨*proof*⟩

instantiation *zhmultiset* :: *ab_group_add*
begin

lift-definition *uminus_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* **is** $\lambda A.\ -A$ ⟨*proof*⟩

lemmas *ZHMSet_uminus* = *uminus_zhmultiset.abs_eq*[*symmetric*]
lemmas *zhmsetmset_uminus* = *uminus_zhmultiset.rep_eq*

instance
⟨*proof*⟩

end

6.5 Infimum and Supremum

instance *zhmultiset* :: *ordered_cancel_comm_monoid_add*
⟨*proof*⟩

instance *zhmultiset* :: *ordered_ab_group_add*
⟨*proof*⟩

instantiation *zhmultiset* :: *distrib_lattice*
begin

definition *inf_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *zhmultiset* **where**
inf_zhmultiset A B = (*if A* < *B then A else B*)

definition *sup_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *zhmultiset* **where**
sup_zhmultiset A B = (*if B* > *A then B else A*)

instance
⟨*proof*⟩

end

end

7 Syntactic Ordinals in Cantor Normal Form

theory *Syntactic_Ordinal*
imports *Hereditary_Multiset* ~~/*src*/*HOL*/*Library*/*Product_Order* ~~/*src*/*HOL*/*Library*/*Extended_Nat*
begin

7.1 Natural (Hessenberg) Product

instantiation *hmultiset* :: *comm_semiring_1*
begin

definition *one_hmultiset* :: *hmultiset* **where**
1 = *HMSet* {#*0*#}

definition *times_hmultiset* :: *hmultiset* ⇒ *hmultiset* ⇒ *hmultiset* **where**
A ∗ *B* = *HMSet* (*image_mset* (*case_prod* (*op* +)) (*hmsetmset A* ×*mset hmsetmset B*))

lemma *hmsetmset_ times*:
hmsetmset (*m* ∗ *n*) = *image_ mset* (*case_ prod* (*op* +)) (*hmsetmset m* ×*mset hmsetmset n*)
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *empty_ times_ left_ hmset*[*simp*]: *HMSet* {#} ∗ *M* = *0*
⟨*proof*⟩

lemma *empty_ times_ right_ hmset*[*simp*]: *M* ∗ *HMSet* {#} = *0*
⟨*proof*⟩

lemma *singleton_ times_ left_ hmset*[*simp*]:
HMSet {#*M*#} ∗ *N* = *HMSet* (*image_ mset* ((*op* +) *M*) (*hmsetmset N*))
⟨*proof*⟩

lemma *singleton_ times_ right_ hmset*[*simp*]:
N ∗ *HMSet* {#*M*#} = *HMSet* (*image_ mset* ((*op* +) *M*) (*hmsetmset N*))
⟨*proof*⟩

7.2 Inequalities

definition *plus_ nmultiset* :: *unit nmultiset* ⇒ *unit nmultiset* ⇒ *unit nmultiset* **where**
plus_ nmultiset X Y = *Rep_ hmultiset* (*Abs_ hmultiset X* + *Abs_ hmultiset Y*)

lemma *plus_ nmultiset_ mono*:
assumes *less*: (*X*, *Y*) < (*X′*, *Y′*) **and** *no_ elem*: *no_ elem X no_ elem Y no_ elem X′ no_ elem Y′*
shows *plus_ nmultiset X Y* < *plus_ nmultiset X′ Y′*
⟨*proof*⟩

lemma *plus_ hmultiset_ transfer*[*transfer_ rule*]:
(*rel_ fun pcr_ hmultiset* (*rel_ fun pcr_ hmultiset pcr_ hmultiset*)) *plus_ nmultiset op* +
⟨*proof*⟩

lemma *Times_ mset_ monoL*:
assumes *less*: *M* < *N* **and** *Z_ nemp*: *Z* ≠ {#}
shows *M* ×*mset Z* < *N* ×*mset Z*
⟨*proof*⟩

lemma *times_ hmultiset_ monoL*:
fixes *a b c* :: *hmultiset*
shows *a* < *b* ⟹ *0* < *c* ⟹ *a* ∗ *c* < *b* ∗ *c*
⟨*proof*⟩

instance *hmultiset* :: *linordered_ semiring_ strict*
⟨*proof*⟩

lemma *zero_ le_ hmset*[*simp*]: *0* ≤ *M* **for** *M* :: *hmultiset*
⟨*proof*⟩

lemma *mult_ le_ mono1_ hmset*: *i* ≤ *j* ⟹ *i* ∗ *k* ≤ *j* ∗ *k* **for** *i j k* :: *hmultiset*
⟨*proof*⟩

lemma *mult_ le_ mono2_ hmset*: *i* ≤ *j* ⟹ *k* ∗ *i* ≤ *k* ∗ *j* **for** *i j k* :: *hmultiset*
⟨*proof*⟩

lemma *mult_ le_ mono_ hmset*: *i* ≤ *j* ⟹ *k* ≤ *l* ⟹ *i* ∗ *k* ≤ *j* ∗ *l* **for** *i j k l* :: *hmultiset*
⟨*proof*⟩

lemma

le_ add1_ hmset: $n \leq n + m$ **and**
le_ add2_ hmset: $n \leq m + n$ **for** n :: *hmultiset*
⟨*proof*⟩

lemma *le_ zero_ eq_ hmset*[*simp*]: $M \leq 0 \longleftrightarrow M = 0$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *not_ less_ zero_ hmset*[*simp*]: $\neg\ M < 0$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *not_ gr_ zero_ hmset*[*simp*]: $\neg\ 0 < M \longleftrightarrow M = 0$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *zero_ less_ iff_ neq_ zero_ hmset*: $0 < M \longleftrightarrow M \neq 0$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *gr_ zeroI_ hmset*: $(M = 0 \Longrightarrow \mathit{False}) \Longrightarrow 0 < M$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *gr_ implies_ not_ zero_ hmset*: $M < N \Longrightarrow N \neq 0$ **for** $M\ N$ :: *hmultiset*
⟨*proof*⟩

lemma *add_ eq_ 0_ iff_ both_ eq_ 0_ hmset*[*simp*]: $M + N = 0 \longleftrightarrow M = 0 \wedge N = 0$ **for** $M\ N$ :: *hmultiset*
⟨*proof*⟩

lemma *zero_ eq_ add_ iff_ both_ eq_ 0_ hmset*[*simp*]: $0 = M + N \longleftrightarrow M = 0 \wedge N = 0$ **for** $M\ N$ :: *hmultiset*
⟨*proof*⟩

lemma *trans_ less_ add1_ hmset*: $i < j \Longrightarrow i < j + m$ **for** $i\ j\ m$ :: *hmultiset*
⟨*proof*⟩

lemma *trans_ less_ add2_ hmset*: $i < j \Longrightarrow i < m + j$ **for** $i\ j\ m$ :: *hmultiset*
⟨*proof*⟩

lemma *trans_ le_ add1_ hmset*: $i \leq j \Longrightarrow i \leq j + m$ **for** $i\ j\ m$ :: *hmultiset*
⟨*proof*⟩

lemma *trans_ le_ add2_ hmset*: $i \leq j \Longrightarrow i \leq m + j$ **for** $i\ j\ m$ :: *hmultiset*
⟨*proof*⟩

lemma *less_ iff_ add1_ le_ hmset*: $m < n \longleftrightarrow m + 1 \leq n$ **for** $m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *zero_ less_ iff_ 1_ le_ hmset*: $0 < n \longleftrightarrow 1 \leq n$ **for** n :: *hmultiset*
⟨*proof*⟩

lemma *less_ add_ 1_ iff_ le_ hmset*: $m < n + 1 \longleftrightarrow m \leq n$ **for** $m\ n$ :: *hmultiset*
⟨*proof*⟩

instance *hmultiset* :: *ordered_ cancel_ comm_ semiring*
⟨*proof*⟩

instance *hmultiset* :: *linordered_ semiring_ 1_ strict*
⟨*proof*⟩

instance *hmultiset* :: *bounded_ lattice_ bot*
⟨*proof*⟩

instance *hmultiset* :: *zero_ less_ one*
⟨*proof*⟩

instance *hmultiset* :: *linordered_ nonzero_ semiring*
⟨*proof*⟩

instance *hmultiset* :: *semiring_no_zero_divisors*
⟨*proof*⟩

lemma *lt_1_iff_eq_0_hmset*: $M < 1 \longleftrightarrow M = 0$ **for** M :: *hmultiset*
⟨*proof*⟩

lemma *zero_less_mult_iff_hmset*[*simp*]: $0 < m * n \longleftrightarrow 0 < m \wedge 0 < n$ **for** $m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *one_le_mult_iff_hmset*[*simp*]: $1 \leq m * n \longleftrightarrow 1 \leq m \wedge 1 \leq n$ **for** $m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_less_cancel2_hmset*[*simp*]: $m * k < n * k \longleftrightarrow 0 < k \wedge m < n$ **for** $k\ m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_less_cancel1_hmset*[*simp*]: $k * m < k * n \longleftrightarrow 0 < k \wedge m < n$ **for** $k\ m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel1_hmset*[*simp*]: $k * m \leq k * n \longleftrightarrow (0 < k \longrightarrow m \leq n)$ **for** $k\ m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel2_hmset*[*simp*]: $m * k \leq n * k \longleftrightarrow (0 < k \longrightarrow m \leq n)$ **for** $k\ m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel_left1_hmset*: $y > 0 \Longrightarrow x \leq x * y$ **for** $x\ y$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel_left2_hmset*: $y \leq 1 \Longrightarrow x * y \leq x$ **for** $x\ y$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel_right1_hmset*: $y > 0 \Longrightarrow x \leq y * x$ **for** $x\ y$ :: *hmultiset*
⟨*proof*⟩

lemma *mult_le_cancel_right2_hmset*: $y \leq 1 \Longrightarrow y * x \leq x$ **for** $x\ y$ :: *hmultiset*
⟨*proof*⟩

lemma *le_square_hmset*: $m \leq m * m$ **for** m :: *hmultiset*
⟨*proof*⟩

lemma *le_cube_hmset*: $m \leq m * (m * m)$ **for** m :: *hmultiset*
⟨*proof*⟩

lemma *diff_le_self_hmset*: $m - n \leq m$ **for** $m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma
less_imp_minus_plus_hmset: $m < n \Longrightarrow k < k - m + n$ **and**
le_imp_minus_plus_hmset: $m \leq n \Longrightarrow k \leq k - m + n$ **for** $k\ m\ n$ :: *hmultiset*
⟨*proof*⟩

lemma *gt_0_lt_mult_gt_1_hmset*:
fixes $m\ n$ :: *hmultiset*
assumes $m > 0$ **and** $n > 1$
shows $m < m * n$
⟨*proof*⟩

instance *hmultiset* :: *linordered_comm_semiring_strict*
⟨*proof*⟩

7.3 Omega

definition ω :: *hmultiset* **where**
$\omega = HMSet\ \{\#1\#\}$

7.4 Embedding of Natural Numbers

lemma *of_nat_hmset*: *of_nat n = HMSet (replicate_mset n 0)*
⟨*proof*⟩

lemma *of_nat_inject_hmset*[*simp*]: *(of_nat m :: hmultiset) = of_nat n ⟷ m = n*
⟨*proof*⟩

lemma *of_nat_minus_hmset*: *of_nat (m − n) = (of_nat m :: hmultiset) − of_nat n*
⟨*proof*⟩

lemma *plus_of_nat_plus_of_nat_hmset*:
k + of_nat m + of_nat n = k + of_nat (m + n) **for** *k :: hmultiset*
⟨*proof*⟩

lemma *plus_of_nat_minus_of_nat_hmset*:
fixes *k :: hmultiset*
assumes *n ≤ m*
shows *k + of_nat m − of_nat n = k + of_nat (m − n)*
⟨*proof*⟩

lemma *of_nat_lt_ω*[*simp*]: *of_nat n < ω*
⟨*proof*⟩

lemma *of_nat_ne_ω*[*simp*]: *of_nat n ≠ ω*
⟨*proof*⟩

lemma *of_nat_less_hmset*[*simp*]: *(of_nat M :: hmultiset) < of_nat N ⟷ M < N*
⟨*proof*⟩

lemma *of_nat_le_hmset*[*simp*]: *(of_nat M :: hmultiset) ≤ of_nat N ⟷ M ≤ N*
⟨*proof*⟩

7.5 Embedding of Extended Natural Numbers

primrec *hmset_of_enat :: enat ⇒ hmultiset* **where**
hmset_of_enat (enat n) = of_nat n
| *hmset_of_enat ∞ = ω*

lemma *hmset_of_enat_0*[*simp*]: *hmset_of_enat 0 = 0*
⟨*proof*⟩

lemma *hmset_of_enat_1*[*simp*]: *hmset_of_enat 1 = 1*
⟨*proof*⟩

lemma *hmset_of_enat_of_nat*[*simp*]: *hmset_of_enat (of_nat n) = of_nat n*
⟨*proof*⟩

lemma *hmset_of_enat_numeral*[*simp*]: *hmset_of_enat (numeral n) = numeral n*
⟨*proof*⟩

lemma *hmset_of_enat_le_ω*[*simp*]: *hmset_of_enat n ≤ ω*
⟨*proof*⟩

lemma *hmset_of_enat_eq_ω_iff*[*simp*]: *hmset_of_enat n = ω ⟷ n = ∞*
⟨*proof*⟩

7.6 Head Omega

definition *head_ω :: hmultiset ⇒ hmultiset* **where**
head_ω M = (if M = 0 then 0 else HMSet {#Max (set_mset (hmsetmset M))#})

lemma *head_ω_eq_0_iff*[*simp*]: *head_ω m = 0 ⟷ m = 0*
⟨*proof*⟩

lemma *head_ω_0*[*simp*]: *head_ω 0 = 0*
⟨*proof*⟩

lemma *head_ω_1*[*simp*]: *head_ω 1 = 1*
⟨*proof*⟩

lemma *head_ω_of_nat*[*simp*]: *head_ω (of_nat n) = (if n = 0 then 0 else 1)*
⟨*proof*⟩

lemma *head_ω_numeral*[*simp*]: *head_ω (numeral n) = 1*
⟨*proof*⟩

lemma *head_ω_ω*[*simp*]: *head_ω ω = ω*
⟨*proof*⟩

lemma *le_imp_head_ω_le*:
assumes *m_le_n*: *m ≤ n*
shows *head_ω m ≤ head_ω n*
⟨*proof*⟩

lemma *head_ω_lt_imp_lt*: *head_ω m < head_ω n ⟹ m < n*
⟨*proof*⟩

lemma *head_ω_plus*[*simp*]: *head_ω (m + n) = sup (head_ω m) (head_ω n)*
⟨*proof*⟩

lemma *head_ω_times*[*simp*]: *head_ω (m * n) = head_ω m * head_ω n*
⟨*proof*⟩

7.7 More Inequalities and Some Equalities

lemma *zero_lt_ω*[*simp*]: *0 < ω*
⟨*proof*⟩

lemma *one_lt_ω*[*simp*]: *1 < ω*
⟨*proof*⟩

lemma *numeral_lt_ω*[*simp*]: *numeral n < ω*
⟨*proof*⟩

lemma *one_le_ω*[*simp*]: *1 ≤ ω*
⟨*proof*⟩

lemma *of_nat_le_ω*[*simp*]: *of_nat n ≤ ω*
⟨*proof*⟩

lemma *numeral_le_ω*[*simp*]: *numeral n ≤ ω*
⟨*proof*⟩

lemma *not_ω_lt_1*[*simp*]: *¬ ω < 1*
⟨*proof*⟩

lemma *not_ω_lt_of_nat*[*simp*]: *¬ ω < of_nat n*
⟨*proof*⟩

lemma *not_ω_lt_numeral*[*simp*]: *¬ ω < numeral n*
⟨*proof*⟩

lemma *not_ω_le_1*[*simp*]: *¬ ω ≤ 1*
⟨*proof*⟩

lemma *not_ω_le_of_nat*[*simp*]: *¬ ω ≤ of_nat n*
⟨*proof*⟩

lemma *not_ω_ le_ numeral*[*simp*]: $\neg\ \omega \leq \mathit{numeral}\ n$
⟨*proof*⟩

lemma *zero_ ne_ ω*[*simp*]: $0 \neq \omega$
⟨*proof*⟩

lemma *one_ ne_ ω*[*simp*]: $1 \neq \omega$
⟨*proof*⟩

lemma *numeral_ ne_ ω*[*simp*]: $\mathit{numeral}\ n \neq \omega$
⟨*proof*⟩

lemma
ω_ ne_ 0[*simp*]: $\omega \neq 0$ **and**
ω_ ne_ 1[*simp*]: $\omega \neq 1$ **and**
ω_ ne_ of_ nat[*simp*]: $\omega \neq \mathit{of\_nat}\ m$ **and**
ω_ ne_ numeral[*simp*]: $\omega \neq \mathit{numeral}\ n$
⟨*proof*⟩

lemma
hmset_ of_ enat_ inject[*simp*]: $\mathit{hmset\_of\_enat}\ m = \mathit{hmset\_of\_enat}\ n \longleftrightarrow m = n$ **and**
hmset_ of_ enat_ less[*simp*]: $\mathit{hmset\_of\_enat}\ m < \mathit{hmset\_of\_enat}\ n \longleftrightarrow m < n$ **and**
hmset_ of_ enat_ le[*simp*]: $\mathit{hmset\_of\_enat}\ m \leq \mathit{hmset\_of\_enat}\ n \longleftrightarrow m \leq n$
⟨*proof*⟩

lemma *lt_ ω_ imp_ ex_ of_ nat*:
assumes *M_ lt_ ω*: $M < \omega$
shows $\exists\, n.\ M = \mathit{of\_nat}\ n$
⟨*proof*⟩

lemma *le_ ω_ imp_ ex_ hmset_ of_ enat*:
assumes *M_ le_ ω*: $M \leq \omega$
shows $\exists\, n.\ M = \mathit{hmset\_of\_enat}\ n$
⟨*proof*⟩

lemma *lt_ ω_ lt_ ω_ imp_ times_ lt_ ω*: $M < \omega \Longrightarrow N < \omega \Longrightarrow M * N < \omega$
⟨*proof*⟩

lemma *times_ ω_ minus_ of_ nat*[*simp*]: $m * \omega - \mathit{of\_nat}\ n = m * \omega$
⟨*proof*⟩

lemma *times_ ω_ minus_ numeral*[*simp*]: $m * \omega - \mathit{numeral}\ n = m * \omega$
⟨*proof*⟩

lemma *ω_ minus_ of_ nat*[*simp*]: $\omega - \mathit{of\_nat}\ n = \omega$
⟨*proof*⟩

lemma *ω_ minus_ 1*[*simp*]: $\omega - 1 = \omega$
⟨*proof*⟩

lemma *ω_ minus_ numeral*[*simp*]: $\omega - \mathit{numeral}\ n = \omega$
⟨*proof*⟩

lemma *hmset_ of_ enat_ minus_ enat*[*simp*]: $\mathit{hmset\_of\_enat}\ (m - \mathit{enat}\ n) = \mathit{hmset\_of\_enat}\ m - \mathit{of\_nat}\ n$
⟨*proof*⟩

lemma *of_ nat_ lt_ hmset_ of_ enat_ iff*: $\mathit{of\_nat}\ m < \mathit{hmset\_of\_enat}\ n \longleftrightarrow \mathit{enat}\ m < n$
⟨*proof*⟩

lemma *of_ nat_ le_ hmset_ of_ enat_ iff*: $\mathit{of\_nat}\ m \leq \mathit{hmset\_of\_enat}\ n \longleftrightarrow \mathit{enat}\ m \leq n$
⟨*proof*⟩

lemma *hmset_of_enat_lt_iff_ne_infinity*: $hmset\_of\_enat\ x < \omega \longleftrightarrow x \neq \infty$
⟨*proof*⟩

lemma *minus_diff_sym_hmset*: $m - (m - n) = n - (n - m)$ **for** $m\ n :: hmultiset$
⟨*proof*⟩

lemma *diff_plus_sym_hmset*: $(c - b) + b = (b - c) + c$ **for** $b\ c :: hmultiset$
⟨*proof*⟩

lemma *times_diff_plus_sym_hmset*: $a * (c - b) + a * b = a * (b - c) + a * c$ **for** $a\ b\ c :: hmultiset$
⟨*proof*⟩

lemma *times_of_nat_minus_left*:
$(of\_nat\ m - of\_nat\ n) * l = of\_nat\ m * l - of\_nat\ n * l$ **for** $l :: hmultiset$
⟨*proof*⟩

lemma *times_of_nat_minus_right*:
$l * (of\_nat\ m - of\_nat\ n) = l * of\_nat\ m - l * of\_nat\ n$ **for** $l :: hmultiset$
⟨*proof*⟩

lemma *lt_ω_imp_times_minus_left*: $m < \omega \Longrightarrow n < \omega \Longrightarrow (m - n) * l = m * l - n * l$
⟨*proof*⟩

lemma *lt_ω_imp_times_minus_right*: $m < \omega \Longrightarrow n < \omega \Longrightarrow l * (m - n) = l * m - l * n$
⟨*proof*⟩

lemma *hmset_pair_decompose*:
$\exists k\ n1\ n2.\ m1 = k + n1 \wedge m2 = k + n2 \wedge (head\_\omega\ n1 \neq head\_\omega\ n2 \vee n1 = 0 \wedge n2 = 0)$
⟨*proof*⟩

lemma *hmset_pair_decompose_less*:
assumes *m1_lt_m2*: $m1 < m2$
shows $\exists k\ n1\ n2.\ m1 = k + n1 \wedge m2 = k + n2 \wedge head\_\omega\ n1 < head\_\omega\ n2$
⟨*proof*⟩

lemma *hmset_pair_decompose_less_eq*:
assumes $m1 \leq m2$
shows $\exists k\ n1\ n2.\ m1 = k + n1 \wedge m2 = k + n2 \wedge (head\_\omega\ n1 < head\_\omega\ n2 \vee n1 = 0 \wedge n2 = 0)$
⟨*proof*⟩

lemma *mono_cross_mult_less_hmset*:
fixes $Aa\ A\ Ba\ B :: hmultiset$
assumes *A_lt*: $A < Aa$ **and** *B_lt*: $B < Ba$
shows $A * Ba + B * Aa < A * B + Aa * Ba$
⟨*proof*⟩

lemma *triple_cross_mult_hmset*:
$An * (Bn * Cn + Bp * Cp - (Bn * Cp + Cn * Bp))$
$+ (Cn * (An * Bp + Bn * Ap - (An * Bn + Ap * Bp))$
$+ (Ap * (Bn * Cp + Cn * Bp - (Bn * Cn + Bp * Cp))$
$+ Cp * (An * Bn + Ap * Bp - (An * Bp + Bn * Ap)))) =$
$An * (Bn * Cp + Cn * Bp - (Bn * Cn + Bp * Cp))$
$+ (Cn * (An * Bn + Ap * Bp - (An * Bp + Bn * Ap))$
$+ (Ap * (Bn * Cn + Bp * Cp - (Bn * Cp + Cn * Bp))$
$+ Cp * (An * Bp + Bn * Ap - (An * Bn + Ap * Bp))))$
for $Ap\ An\ Bp\ Bn\ Cp\ Cn\ Dp\ Dn :: hmultiset$
⟨*proof*⟩

7.8 Conversions to Natural Numbers

definition *offset_hmset* :: $hmultiset \Rightarrow nat$ **where**
$offset\_hmset\ M = count\ (hmsetmset\ M)\ 0$

lemma *offset_hmset_of_nat*[*simp*]: $offset\_hmset\ (of\_nat\ n) = n$

⟨*proof*⟩

lemma *offset_hmset_numeral*[*simp*]: *offset_hmset* (*numeral n*) = *numeral n*
⟨*proof*⟩

definition *sum_coefs* :: *hmultiset* ⇒ *nat* **where**
sum_coefs M = *size* (*hmsetmset M*)

lemma *sum_coefs_distrib_plus*[*simp*]: *sum_coefs* (*M* + *N*) = *sum_coefs M* + *sum_coefs N*
⟨*proof*⟩

lemma *sum_coefs_gt_0*: *sum_coefs M* > 0 ⟷ *M* > 0
⟨*proof*⟩

7.9 An Example

The following proof is based on an informal proof by Uwe Waldmann, inspired by a similar argument by Michel Ludwig.

lemma *waldmann_less*:
fixes *α1 α2 β1 β2 γ δ* :: *hmultiset*
assumes
αβ2γ_lt_αβ1γ: *α2* + *β2* ∗ *γ* < *α1* + *β1* ∗ *γ* **and**
β2_le_β1: *β2* ≤ *β1* **and**
γ_lt_δ: *γ* < *δ*
shows *α2* + *β2* ∗ *δ* < *α1* + *β1* ∗ *δ*
⟨*proof*⟩

end

8 Signed Syntactic Ordinals in Cantor Normal Form

theory *Signed_Syntactic_Ordinal*
imports *Signed_Hereditary_Multiset Syntactic_Ordinal*
begin

8.1 Natural (Hessenberg) Product

instantiation *zhmultiset* :: *comm_ring_1*
begin

lift-definition *one_zhmultiset* :: *zhmultiset* **is** {#*0*#}$_z$ ⟨*proof*⟩

lift-definition *times_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* ⇒ *zhmultiset* **is**
λM N.
zmset_of (*hmsetmset* (*HMSet* (*mset_pos M*) ∗ *HMSet* (*mset_pos N*)))
− *zmset_of* (*hmsetmset* (*HMSet* (*mset_pos M*) ∗ *HMSet* (*mset_neg N*)))
+ *zmset_of* (*hmsetmset* (*HMSet* (*mset_neg M*) ∗ *HMSet* (*mset_neg N*)))
− *zmset_of* (*hmsetmset* (*HMSet* (*mset_neg M*) ∗ *HMSet* (*mset_pos N*))) ⟨*proof*⟩

lemmas *zhmsetmset_times* = *times_zhmultiset.rep_eq*

instance
⟨*proof*⟩

end

lemma *zhmset_of_1*: *zhmset_of 1* = *1*
⟨*proof*⟩

lemma *zhmset_of_times*: *zhmset_of* (*A* ∗ *B*) = *zhmset_of A* ∗ *zhmset_of B*
⟨*proof*⟩

lemma *zhmset_of_prod_list*:
zhmset_of (*prod_list Ms*) = *prod_list* (*map zhmset_of Ms*)
⟨*proof*⟩

8.2 Omega

definition ω_z :: *zhmultiset* **where**
ω_z = *ZHMSet* $\{\#1\#\}_z$

lemma ω_z*_as_*ω: ω_z = *zhmset_of* ω
⟨*proof*⟩

8.3 Embedding of Natural Numbers

lemma *of_nat_zhmset*: *of_nat n* = *zhmset_of* (*of_nat n*)
⟨*proof*⟩

lemma *of_nat_inject_zhmset*[*simp*]: (*of_nat m* :: *zhmultiset*) = *of_nat n* ⟷ *m* = *n*
⟨*proof*⟩

lemma *plus_of_nat_plus_of_nat_zhmset*:
k + *of_nat m* + *of_nat n* = *k* + *of_nat* (*m* + *n*) **for** *k* :: *zhmultiset*
⟨*proof*⟩

lemma *plus_of_nat_minus_of_nat_zhmset*:
fixes *k* :: *zhmultiset*
assumes *n* ≤ *m*
shows *k* + *of_nat m* − *of_nat n* = *k* + *of_nat* (*m* − *n*)
⟨*proof*⟩

lemma *of_nat_lt_*ω_z[*simp*]: *of_nat n* < ω_z
⟨*proof*⟩

lemma *of_nat_ne_*ω_z[*simp*]: *of_nat n* ≠ ω_z
⟨*proof*⟩

8.4 Embedding of Extended Natural Numbers

primrec *zhmset_of_enat* :: *enat* ⇒ *zhmultiset* **where**
zhmset_of_enat (*enat n*) = *of_nat n*
| *zhmset_of_enat* ∞ = ω_z

lemma *zhmset_of_enat_0*[*simp*]: *zhmset_of_enat 0* = *0*
⟨*proof*⟩

lemma *zhmset_of_enat_1*[*simp*]: *zhmset_of_enat 1* = *1*
⟨*proof*⟩

lemma *zhmset_of_enat_of_nat*[*simp*]: *zhmset_of_enat* (*of_nat n*) = *of_nat n*
⟨*proof*⟩

lemma *zhmset_of_enat_numeral*[*simp*]: *zhmset_of_enat* (*numeral n*) = *numeral n*
⟨*proof*⟩

lemma *zhmset_of_enat_le_*ω_z[*simp*]: *zhmset_of_enat n* ≤ ω_z
⟨*proof*⟩

lemma *zhmset_of_enat_eq_*ω_z*_iff*[*simp*]: *zhmset_of_enat n* = ω_z ⟷ *n* = ∞
⟨*proof*⟩

8.5 Inequalities and Some (Dis)equalities

instance *zhmultiset* :: *zero_less_one*
⟨*proof*⟩

instantiation *zhmultiset* :: *linordered_idom*
begin

definition *sgn_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* **where**
sgn_zhmultiset M = (*if M* = *0 then 0 else if M* > *0 then 1 else* −*1*)

definition *abs_zhmultiset* :: *zhmultiset* ⇒ *zhmultiset* **where**
abs_zhmultiset M = (*if M* < *0 then* − *M else M*)

lemma *gt_0_times_gt_0_imp*:
fixes *a b* :: *zhmultiset*
assumes *a_gt0*: *a* > *0* **and** *b_gt0*: *b* > *0*
shows *a* ∗ *b* > *0*
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *le_zhmset_of_pos*: *M* ≤ *zhmset_of* (*hmset_pos M*)
⟨*proof*⟩

lemma *minus_zhmset_of_pos_le*: − *zhmset_of* (*hmset_neg M*) ≤ *M*
⟨*proof*⟩

lemma *zhmset_of_nonneg*[*simp*]: *zhmset_of M* ≥ *0*
⟨*proof*⟩

lemma
fixes *n* :: *zhmultiset*
assumes *0* ≤ *m*
shows
le_add1_hmset: *n* ≤ *n* + *m* **and**
le_add2_hmset: *n* ≤ *m* + *n*
⟨*proof*⟩

lemma *less_iff_add1_le_zhmset*: *m* < *n* ⟷ *m* + *1* ≤ *n* **for** *m n* :: *zhmultiset*
⟨*proof*⟩

lemma *gt_0_lt_mult_gt_1_zhmset*:
fixes *m n* :: *zhmultiset*
assumes *m* > *0* **and** *n* > *1*
shows *m* < *m* ∗ *n*
⟨*proof*⟩

lemma *zero_less_iff_1_le_zhmset*: *0* < *n* ⟷ *1* ≤ *n* **for** *n* :: *zhmultiset*
⟨*proof*⟩

lemma *less_add_1_iff_le_hmset*: *m* < *n* + *1* ⟷ *m* ≤ *n* **for** *m n* :: *zhmultiset*
⟨*proof*⟩

lemma *nonneg_le_mult_right_mono_zhmset*:
fixes *x y z* :: *zhmultiset*
assumes *x*: *0* ≤ *x* **and** *y*: *0* < *y* **and** *z*: *x* ≤ *z*
shows *x* ≤ *y* ∗ *z*
⟨*proof*⟩

instance *hmultiset* :: *ordered_cancel_comm_semiring*
⟨*proof*⟩

instance *hmultiset* :: *linordered_semiring_1_strict*

⟨*proof*⟩

instance *hmultiset* :: *bounded_lattice_bot*
⟨*proof*⟩

instance *hmultiset* :: *zero_less_one*
⟨*proof*⟩

instance *hmultiset* :: *linordered_nonzero_semiring*
⟨*proof*⟩

instance *hmultiset* :: *semiring_no_zero_divisors*
⟨*proof*⟩

lemma *zero_lt_ω_z*[*simp*]: $0 < \omega_z$
⟨*proof*⟩

lemma *one_lt_ω*[*simp*]: $1 < \omega_z$
⟨*proof*⟩

lemma *numeral_lt_ω_z*[*simp*]: *numeral* $n < \omega_z$
⟨*proof*⟩

lemma *one_le_ω_z*[*simp*]: $1 \leq \omega_z$
⟨*proof*⟩

lemma *of_nat_le_ω_z*[*simp*]: *of_nat* $n \leq \omega_z$
⟨*proof*⟩

lemma *numeral_le_ω_z*[*simp*]: *numeral* $n \leq \omega_z$
⟨*proof*⟩

lemma *not_ω_z_lt_1*[*simp*]: $\neg\ \omega_z < 1$
⟨*proof*⟩

lemma *not_ω_z_lt_of_nat*[*simp*]: $\neg\ \omega_z <$ *of_nat* n
⟨*proof*⟩

lemma *not_ω_z_lt_numeral*[*simp*]: $\neg\ \omega_z <$ *numeral* n
⟨*proof*⟩

lemma *not_ω_z_le_1*[*simp*]: $\neg\ \omega_z \leq 1$
⟨*proof*⟩

lemma *not_ω_z_le_of_nat*[*simp*]: $\neg\ \omega_z \leq$ *of_nat* n
⟨*proof*⟩

lemma *not_ω_z_le_numeral*[*simp*]: $\neg\ \omega_z \leq$ *numeral* n
⟨*proof*⟩

lemma *zero_ne_ω_z*[*simp*]: $0 \neq \omega_z$
⟨*proof*⟩

lemma *one_ne_ω_z*[*simp*]: $1 \neq \omega_z$
⟨*proof*⟩

lemma *numeral_ne_ω_z*[*simp*]: *numeral* $n \neq \omega_z$
⟨*proof*⟩

lemma
ω_z_ne_0[*simp*]: $\omega_z \neq 0$ **and**
ω_z_ne_1[*simp*]: $\omega_z \neq 1$ **and**
ω_z_ne_of_nat[*simp*]: $\omega_z \neq$ *of_nat* m **and**

$\omega_z\_ne\_numeral[simp]$: $\omega_z \neq numeral\ n$
⟨*proof*⟩

lemma
$zhmset\_of\_enat\_inject[simp]$: $zhmset\_of\_enat\ m = zhmset\_of\_enat\ n \longleftrightarrow m = n$ **and**
$zhmset\_of\_enat\_lt\_iff\_lt[simp]$: $zhmset\_of\_enat\ m < zhmset\_of\_enat\ n \longleftrightarrow m < n$ **and**
$zhmset\_of\_enat\_le\_iff\_le[simp]$: $zhmset\_of\_enat\ m \leq zhmset\_of\_enat\ n \longleftrightarrow m \leq n$
⟨*proof*⟩

lemma $of\_nat\_lt\_zhmset\_of\_enat\_iff$: $of\_nat\ m < zhmset\_of\_enat\ n \longleftrightarrow enat\ m < n$
⟨*proof*⟩

lemma $of\_nat\_le\_zhmset\_of\_enat\_iff$: $of\_nat\ m \leq zhmset\_of\_enat\ n \longleftrightarrow enat\ m \leq n$
⟨*proof*⟩

lemma $zhmset\_of\_enat\_lt\_iff\_ne\_infinity$: $zhmset\_of\_enat\ x < \omega_z \longleftrightarrow x \neq \infty$
⟨*proof*⟩

8.6 An Example

A new proof of $[\![?\alpha 2.0 + ?\beta 2.0 * ?\gamma < ?\alpha 1.0 + ?\beta 1.0 * ?\gamma;\ ?\beta 2.0 \leq ?\beta 1.0;\ ?\gamma < ?\delta]\!] \Longrightarrow ?\alpha 2.0 + ?\beta 2.0 * ?\delta < ?\alpha 1.0 + ?\beta 1.0 * ?\delta$:

lemma
fixes $\alpha 1\ \alpha 2\ \beta 1\ \beta 2\ \gamma\ \delta$ :: *hmultiset*
assumes
$\alpha\beta 2\gamma\_lt\_\alpha\beta 1\gamma$: $\alpha 2 + \beta 2 * \gamma < \alpha 1 + \beta 1 * \gamma$ **and**
$\beta 2\_le\_\beta 1$: $\beta 2 \leq \beta 1$ **and**
γ_lt_δ: $\gamma < \delta$
shows $\alpha 2 + \beta 2 * \delta < \alpha 1 + \beta 1 * \delta$
⟨*proof*⟩

end