A Partition Theorem for the Ordinal ω^ω

Lawrence C. Paulson

June 17, 2024

Abstract

The theory of partition relations concerns generalisations of Ramsey's theorem. For any ordinal α, write $\alpha \to (\alpha, m)^2$ if for each function f from unordered pairs of elements of α into $\{0,1\}$, either there is a subset $X \subseteq \alpha$ order-isomorphic to α such that $f\{x,y\} = 0$ for all $\{x,y\} \subseteq X$, or there is an m element set $Y \subseteq \alpha$ such that $f\{x,y\} = 1$ for all $\{x,y\} \subseteq Y$. (In both cases, with $\{x,y\}$ we require $x \neq y$.) In particular, the infinite Ramsey theorem can be written in this notation as $\omega \to (\omega, \omega)^2$, or if we restrict m to the positive integers as above, then $\omega \to (\omega, m)^2$ for all m [3].

This entry formalises Larson's proof of $\omega^\omega \to (\omega^\omega, m)^2$ along with a similar proof of a result due to Specker: $\omega^2 \to (\omega^2, m)^2$. Also proved is a necessary result by Erdős and Milner [1, 2]: $\omega^{1+\alpha\cdot n} \to (\omega^{1+\alpha}, 2^n)^2$.

These examples demonstrate the use of Isabelle/HOL to formalise advanced results that combine ZF set theory with basic concepts like lists and natural numbers.

Contents

1 Library additions

theory *Library-Additions*
imports *ZFC-in-HOL.Ordinal-Exp HOL−Library.Ramsey Nash-Williams.Nash-Williams*

begin

lemma *finite-enumerate-Diff-singleton*:
fixes $S :: 'a::wellorder\ set$
assumes *finite S* **and** i: $i < card\ S\ enumerate\ S\ i < x$
shows $enumerate\ (S - \{x\})\ i = enumerate\ S\ i$
⟨*proof*⟩

lemma *hd-lex*: $[\![hd\ ms < hd\ ns;\ length\ ms = length\ ns;\ ns \neq []]\!] \Longrightarrow (ms, ns) \in lex\ less\text{-}than$
⟨*proof*⟩

lemma *sorted-hd-le*:
assumes $sorted\ xs\ x \in list.set\ xs$
shows $hd\ xs \leq x$
⟨*proof*⟩

lemma *sorted-le-last*:

assumes *sorted xs* $x \in$ *list.set xs*
shows $x \leq$ *last xs*
⟨*proof*⟩

lemma *hd-list-of*:
assumes *finite A* $A \neq \{\}$
shows *hd* (*sorted-list-of-set A*) = *Min A*
⟨*proof*⟩

lemma *sorted-hd-le-last*:
assumes *sorted xs* $xs \neq []$
shows *hd xs* $\leq$ *last xs*
⟨*proof*⟩

lemma *sorted-list-of-set-set-of* [*simp*]: *strict-sorted l* $\Longrightarrow$ *sorted-list-of-set* (*list.set l*) = *l*
⟨*proof*⟩

lemma *range-strict-mono-ext*:
fixes *f*::*nat* ⇒ *'a*::*linorder*
assumes *eq*: *range f* = *range g*
and *sm*: *strict-mono f strict-mono g*
shows *f* = *g*
⟨*proof*⟩

1.1 Other material

definition *strict-mono-sets* :: [*'a*::*order set*, *'a*::*order* ⇒ *'b*::*order set*] ⇒ *bool* **where**
strict-mono-sets A f ≡ ∀ *x*∈*A*. ∀ *y*∈*A*. $x < y \longrightarrow$ *less-sets* (*f x*) (*f y*)

lemma *strict-mono-setsD*:
assumes *strict-mono-sets A f* $x < y$ $x \in A$ $y \in A$
shows *less-sets* (*f x*) (*f y*)
⟨*proof*⟩

lemma *strict-mono-sets-imp-disjoint*:
fixes *A* :: *'a*::*linorder set*
assumes *strict-mono-sets A f*
shows *pairwise* (λ*x y*. *disjnt* (*f x*) (*f y*)) *A*
⟨*proof*⟩

lemma *strict-mono-sets-subset*:
assumes *strict-mono-sets B f* $A \subseteq B$
shows *strict-mono-sets A f*
⟨*proof*⟩

lemma *strict-mono-less-sets-Min*:
assumes *strict-mono-sets I f finite I* $I \neq \{\}$
shows *less-sets* (*f* (*Min I*)) (⋃ (*f* ‘ (*I* − {*Min I*})))

$\langle proof \rangle$

lemma *pair-less-iff1* [*simp*]: $((x,y),\ (x,z)) \in pair\text{-}less \longleftrightarrow y<z$
$\langle proof \rangle$

lemma *infinite-finite-Inter*:
assumes $finite\ \mathcal{A}\ \mathcal{A}\neq\{\}\ \bigwedge A.\ A \in \mathcal{A} \Longrightarrow infinite\ A$
and $\bigwedge A\ B.\ [\![A \in \mathcal{A};\ B \in \mathcal{A}]\!] \Longrightarrow A \cap B \in \mathcal{A}$
shows $infinite\ (\bigcap \mathcal{A})$
$\langle proof \rangle$

lemma *atLeast-less-sets*: $[\![less\text{-}sets\ A\ \{x\};\ B \subseteq \{x..\}]\!] \Longrightarrow less\text{-}sets\ A\ B$
$\langle proof \rangle$

1.2 The list-of function

lemma *sorted-list-of-set-insert-remove-cons*:
assumes $finite\ A\ less\text{-}sets\ \{a\}\ A$
shows $sorted\text{-}list\text{-}of\text{-}set\ (insert\ a\ A) = a\ \#\ sorted\text{-}list\text{-}of\text{-}set\ A$
$\langle proof \rangle$

lemma *sorted-list-of-set-Un*:
assumes *AB*: $less\text{-}sets\ A\ B$ **and** *fin*: $finite\ A\ finite\ B$
shows $sorted\text{-}list\text{-}of\text{-}set\ (A \cup B) = sorted\text{-}list\text{-}of\text{-}set\ A\ @\ sorted\text{-}list\text{-}of\text{-}set\ B$
$\langle proof \rangle$

lemma *sorted-list-of-set-UN-lessThan*:
fixes $k{::}nat$
assumes *sm*: $strict\text{-}mono\text{-}sets\ \{..<k\}\ A$ **and** $\bigwedge i.\ i < k \Longrightarrow finite\ (A\ i)$
shows $sorted\text{-}list\text{-}of\text{-}set\ (\bigcup i<k.\ A\ i) = concat\ (map\ (sorted\text{-}list\text{-}of\text{-}set \circ A)\ (sorted\text{-}list\text{-}of\text{-}set\ \{..<k\}))$
$\langle proof \rangle$

lemma *sorted-list-of-set-UN-atMost*:
fixes $k{::}nat$
assumes $strict\text{-}mono\text{-}sets\ \{..k\}\ A$ **and** $\bigwedge i.\ i \le k \Longrightarrow finite\ (A\ i)$
shows $sorted\text{-}list\text{-}of\text{-}set\ (\bigcup i\le k.\ A\ i) = concat\ (map\ (sorted\text{-}list\text{-}of\text{-}set \circ A)\ (sorted\text{-}list\text{-}of\text{-}set\ \{..k\}))$
$\langle proof \rangle$

1.3 Monotonic enumeration of a countably infinite set

abbreviation $enum \equiv enumerate$

Could be generalised to infinite countable sets of any type

lemma *nat-infinite-iff*:
fixes $N :: nat\ set$
shows $infinite\ N \longleftrightarrow (\exists f{::}nat\Rightarrow nat.\ N = range\ f \wedge strict\text{-}mono\ f)$
$\langle proof \rangle$

lemma *enum-works*:
  fixes N :: *nat set*
  assumes *infinite* N
  shows N = *range* (*enum* N) $\wedge$ *strict-mono* (*enum* N)
  ⟨*proof*⟩

lemma *range-enum*: *range* (*enum* N) = N **and** *strict-mono-enum*: *strict-mono* (*enum* N)
  if *infinite* N **for** N :: *nat set*
  ⟨*proof*⟩

lemma *enum-0-eq-Inf*:
  fixes N :: *nat set*
  assumes *infinite* N
  shows *enum* N *0* = *Inf* N
⟨*proof*⟩

lemma *enum-works-finite*:
  fixes N :: *nat set*
  assumes *finite* N
  shows N = *enum* N ‘ {..<*card* N} $\wedge$ *strict-mono-on* {..<*card* N} (*enum* N)
  ⟨*proof*⟩

lemma *enum-obtain-index-finite*:
  fixes N :: *nat set*
  assumes $x \in N$ *finite* N
  obtains i **where** i < *card* N x = *enum* N i
  ⟨*proof*⟩

lemma *enum-0-eq-Inf-finite*:
  fixes N :: *nat set*
  assumes *finite* N $N \neq \{\}$
  shows *enum* N *0* = *Inf* N
⟨*proof*⟩

lemma *greaterThan-less-enum*:
  fixes N :: *nat set*
  assumes $N \subseteq \{x<..\}$ *infinite* N
  shows x < *enum* N i
  ⟨*proof*⟩

lemma *atLeast-le-enum*:
  fixes N :: *nat set*
  assumes $N \subseteq \{x..\}$ *infinite* N
  shows $x \leq$ *enum* N i
  ⟨*proof*⟩

lemma *less-sets-empty1* [*simp*]: *less-sets* {} A **and** *less-sets-empty2* [*simp*]: *less-sets* A {}

$\langle proof \rangle$

lemma *less-sets-singleton1* [*simp*]: *less-sets* $\{a\}$ $A \longleftrightarrow (\forall x{\in}A.\ a < x)$
and *less-sets-singleton2* [*simp*]: *less-sets* A $\{a\} \longleftrightarrow (\forall x{\in}A.\ x < a)$
$\langle proof \rangle$

lemma *less-sets-atMost* [*simp*]: *less-sets* $\{..a\}$ $A \longleftrightarrow (\forall x{\in}A.\ a < x)$
and *less-sets-alLeast* [*simp*]: *less-sets* A $\{a..\} \longleftrightarrow (\forall x{\in}A.\ x < a)$
$\langle proof \rangle$

lemma *less-sets-imp-strict-mono-sets*:
assumes $\bigwedge i.$ *less-sets* $(A\ i)$ $(A\ (Suc\ i))$ $\bigwedge i.\ i{>}0 \Longrightarrow A\ i \neq \{\}$
shows *strict-mono-sets UNIV A*
$\langle proof \rangle$

lemma *less-sets-Suc-Max*:
assumes *finite A*
shows *less-sets* A $\{Suc\ (Max\ A)..\}$
$\langle proof \rangle$

lemma *infinite-nat-greaterThan*:
fixes $m{::}nat$
assumes *infinite N*
shows *infinite* $(N \cap \{m{<}..\})$
$\langle proof \rangle$

end

2 Ordinal Partitions

Material from Jean A. Larson, A short proof of a partition theorem for the ordinal ω^ω. *Annals of Mathematical Logic*, 6:129–145, 1973. Also from "Partition Relations" by A. Hajnal and J. A. Larson, in *Handbook of Set Theory*, edited by Matthew Foreman and Akihiro Kanamori (Springer, 2010).

theory *Partitions*
imports *Library-Additions ZFC-in-HOL.ZFC-Typeclasses ZFC-in-HOL.Cantor-NF*

begin

abbreviation $tp :: V\ set \Rightarrow V$
where $tp\ A \equiv ordertype\ A\ VWF$

2.1 Ordinal Partitions: Definitions

definition *partn-lst* :: $[('a \times 'a)\ set,\ 'a\ set,\ V\ list,\ nat] \Rightarrow bool$
where $partn\text{-}lst\ r\ B\ \alpha\ n \equiv \forall f \in [B]^n \ \rightarrow\ \{..{<}length\ \alpha\}.$
$\exists\, i < length\ \alpha.\ \exists\, H.\ H \subseteq B \wedge ordertype\ H\ r = (\alpha!i) \wedge f\ \text{`}\ (nsets\ H\ n)$
$\subseteq \{i\}$

abbreviation *partn-lst-VWF* :: $V \Rightarrow V$ *list* $\Rightarrow$ *nat* $\Rightarrow$ *bool*
where *partn-lst-VWF* $\beta \equiv$ *partn-lst VWF* (*elts* β)

lemma *partn-lst-E*:
assumes *partn-lst r B* α *n f* $\in$ *nsets B n* $\rightarrow$ $\{..<l\}$ $l =$ *length* α
obtains *i H* **where** $i < l$ $H \subseteq B$
ordertype H r $= \alpha!i$ *f* ' (*nsets H n*) $\subseteq \{i\}$
$\langle proof \rangle$

lemma *partn-lst-VWF-nontriv*:
assumes *partn-lst-VWF* β α *n* $l =$ *length* α *Ord* β $l > 0$
obtains *i* **where** $i < l$ $\alpha!i \leq \beta$
$\langle proof \rangle$

lemma *partn-lst-triv0*:
assumes $\alpha!i = 0$ $i <$ *length* α $n \neq 0$
shows *partn-lst r B* α *n*
$\langle proof \rangle$

lemma *partn-lst-triv1*:
assumes $\alpha!i \leq 1$ $i <$ *length* α $n > 1$ $B \neq \{\}$ *wf r*
shows *partn-lst r B* α *n*
$\langle proof \rangle$

lemma *partn-lst-two-swap*:
assumes *partn-lst r B* $[x,y]$ *n* **shows** *partn-lst r B* $[y,x]$ *n*
$\langle proof \rangle$

lemma *partn-lst-greater-resource*:
assumes *M*: *partn-lst r B* α *n* **and** $B \subseteq C$
shows *partn-lst r C* α *n*
$\langle proof \rangle$

lemma *partn-lst-less*:
assumes *M*: *partn-lst r B* α *n* **and** *eq*: *length* $\alpha' =$ *length* α **and** *List.set* $\alpha' \subseteq$ *ON*
and *le*: $\bigwedge i.\ i <$ *length* $\alpha \Longrightarrow \alpha'!i \leq \alpha!i$
and *r*: *wf r trans r total-on B r* **and** *small B*
shows *partn-lst r B* α' *n*
$\langle proof \rangle$

Holds because no n-sets exist!

lemma *partn-lst-VWF-degenerate*:
assumes $k < n$
shows *partn-lst-VWF* ω (*ord-of-nat k* # αs) *n*
$\langle proof \rangle$

lemma *partn-lst-VWF-ω-2*:
  assumes *Ord* α
  shows *partn-lst-VWF* $(\omega \uparrow (1+\alpha))$ $[2, \omega \uparrow (1+\alpha)]$ 2 (**is** *partn-lst-VWF ?β - -*)
⟨*proof*⟩

2.2 Relating partition properties on *VWF* to the general case

Two very similar proofs here!

lemma *partn-lst-imp-partn-lst-VWF-eq*:
  assumes *part*: *partn-lst r U* α *n* **and** β: *ordertype U r* $= \beta$ **and** *small U*
    and *r*: *wf r trans r total-on U r*
  shows *partn-lst-VWF* β α *n*
  ⟨*proof*⟩

lemma *partn-lst-imp-partn-lst-VWF*:
  assumes *part*: *partn-lst r U* α *n* **and** β: *ordertype U r* $\leq \beta$ *small U*
    and *r*: *wf r trans r total-on U r*
  shows *partn-lst-VWF* β α *n*
  ⟨*proof*⟩

lemma *partn-lst-VWF-imp-partn-lst-eq*:
  assumes *part*: *partn-lst-VWF* β α *n* **and** β: *ordertype U r* $= \beta$ *small U*
    and *r*: *wf r trans r total-on U r*
  shows *partn-lst r U* α *n*
  ⟨*proof*⟩

corollary *partn-lst-VWF-imp-partn-lst*:
  assumes *partn-lst-VWF* β α *n* **and** β: *ordertype U r* $\geq \beta$ *small U*
          wf r trans r total-on U r
  shows *partn-lst r U* α *n*
  ⟨*proof*⟩

2.3 Simple consequences of the definitions

A restatement of the infinite Ramsey theorem using partition notation

lemma *Ramsey-partn*: *partn-lst-VWF* ω $[\omega,\omega]$ 2
⟨*proof*⟩

This is the counterexample sketched in Hajnal and Larson, section 9.1.

proposition *omega-basic-counterexample*:
  assumes *Ord* α
  shows $\neg$ *partn-lst-VWF* α [*succ* (*vcard* α), ω] 2
⟨*proof*⟩

2.4 Specker's theorem

definition *form-split* :: [*nat*,*nat*,*nat*,*nat*,*nat*] $\Rightarrow$ *bool* **where**
  form-split a b c d i $\equiv a \leq c \wedge (i=0 \wedge a<b \wedge b<c \wedge c<d \vee$
                    $i=1 \wedge a<c \wedge c<b \wedge b<d \vee$

$i=2 \wedge a<c \wedge c<d \wedge d<b \vee$
$i=3 \wedge a=c \wedge b \neq d)$

definition *form* :: $[(nat*nat)set,\ nat] \Rightarrow bool$ **where**
form u i $\equiv \exists\, a\ b\ c\ d.\ u = \{(a,b),(c,d)\} \wedge$ *form-split a b c d i*

definition *scheme* :: $[(nat*nat)set] \Rightarrow nat\ set$ **where**
scheme u $\equiv$ *fst* ‘ $u \cup$ *snd* ‘ u

definition *UU* :: $(nat*nat)\ set$
where $UU \equiv \{(a,b).\ a < b\}$

lemma *ordertype-UNIV-ω2*: *ordertype UNIV pair-less* $= \omega\uparrow 2$
⟨*proof*⟩

lemma *ordertype-UU-ge-ω2*: *ordertype UNIV pair-less* $\leq$ *ordertype UU pair-less*
⟨*proof*⟩

lemma *ordertype-UU-ω2*: *ordertype UU pair-less* $= \omega\uparrow 2$
⟨*proof*⟩

Lemma 2.3 of Jean A. Larson, A short proof of a partition theorem for the ordinal ω^ω. *Annals of Mathematical Logic*, 6:129–145, 1973.

lemma *lemma-2-3*:
fixes $f :: (nat \times nat)\ set \Rightarrow nat$
assumes $f \in [UU]^2 \rightarrow \{..<Suc\ (Suc\ 0)\}$
obtains *N js* **where** *infinite N* **and** $\bigwedge k\ u.\ [\![k < 4;\ u \in [UU]^2;$ *form u k*; *scheme* $u \subseteq N]\!] \Longrightarrow f\ u = js!k$
⟨*proof*⟩

Lemma 2.4 of Jean A. Larson, ibid.

lemma *lemma-2-4*:
assumes *infinite N* $k < 4$
obtains *M* **where** $M \in [UU]^m$ $\bigwedge u.\ u \in [M]^2 \Longrightarrow$ *form u k* $\bigwedge u.\ u \in [M]^2 \Longrightarrow$ *scheme* $u \subseteq N$
⟨*proof*⟩

Lemma 2.5 of Jean A. Larson, ibid.

lemma *lemma-2-5*:
assumes *infinite N*
obtains *X* **where** $X \subseteq UU$ *ordertype X pair-less* $= \omega\uparrow 2$
$\bigwedge u.\ u \in [X]^2 \Longrightarrow (\exists\, k<4.$ *form u k*$) \wedge$ *scheme* $u \subseteq N$
⟨*proof*⟩

Theorem 2.1 of Jean A. Larson, ibid.

lemma *Specker-aux*:
assumes $\alpha \in$ *elts* ω
shows *partn-lst pair-less UU* $[\omega\uparrow 2,\alpha]$ *2*
⟨*proof*⟩

theorem *Specker*: $\alpha \in elts\ \omega \Longrightarrow partn\text{-}lst\text{-}VWF\ (\omega\uparrow 2)\ [\omega\uparrow 2,\alpha]\ 2$
⟨*proof*⟩

end
theory *Erdos-Milner*
imports *Partitions*

begin

2.5 Erds-Milner theorem

P. Erds and E. C. Milner, A Theorem in the Partition Calculus. Canadian Math. Bull. 15:4 (1972), 501-505. Corrigendum, Canadian Math. Bull. 17:2 (1974), 305.

The paper defines strong types as satisfying the criteria below. It remarks that ordinals satisfy those criteria. Here is a (too complicated) proof.

proposition *strong-ordertype-eq*:
assumes D: $D \subseteq elts\ \beta$ **and** $Ord\ \beta$
obtains L **where** $\bigcup(List.set\ L) = D$ $\bigwedge X.\ X \in List.set\ L \Longrightarrow indecomposable\ (tp\ X)$
and $\bigwedge M.\ [\![M \subseteq D;\ \bigwedge X.\ X \in List.set\ L \Longrightarrow tp\ (M \cap X) \geq tp\ X]\!] \Longrightarrow tp\ M = tp\ D$
⟨*proof*⟩

The "remark" of Erds and E. C. Milner, Canad. Math. Bull. Vol. 17 (2), 1974

proposition *indecomposable-imp-Ex-less-sets*:
assumes *indec*: $indecomposable\ \alpha$ **and** $\alpha \geq \omega$
and A: $tp\ A = \alpha\ small\ A\ A \subseteq ON$
and $x \in A$ **and** $A1$: $tp\ A1 = \alpha\ A1 \subseteq A$
obtains $A2$ **where** $tp\ A2 = \alpha\ A2 \subseteq A1\ \{x\} \ll A2$
⟨*proof*⟩

the main theorem, from which they derive the headline result

theorem *Erdos-Milner-aux*:
assumes *part*: $partn\text{-}lst\text{-}VWF\ \alpha\ [k,\ \gamma]\ 2$
and *indec*: $indecomposable\ \alpha$ **and** $k > 1\ Ord\ \gamma$ **and** β: $\beta \in elts\ \omega 1$
shows $partn\text{-}lst\text{-}VWF\ (\alpha{*}\beta)\ [ord\text{-}of\text{-}nat\ (2{*}k),\ min\ \gamma\ (\omega{*}\beta)]\ 2$
⟨*proof*⟩

theorem *Erdos-Milner*:
assumes ν: $\nu \in elts\ \omega 1$
shows $partn\text{-}lst\text{-}VWF\ (\omega\uparrow(1 + \nu * n))\ [ord\text{-}of\text{-}nat\ (2\hat{\ }n),\ \omega\uparrow(1{+}\nu)]\ 2$
⟨*proof*⟩

corollary *remark-3*: *partn-lst-VWF* $(\omega\uparrow(Suc(4*k)))$ $[4, \omega\uparrow(Suc(2*k))]$ *2*
⟨*proof*⟩

Theorem 3.2 of Jean A. Larson, ibid.

corollary *Theorem-3-2*:
fixes *k n::nat*
shows *partn-lst-VWF* $(\omega\uparrow(n*k))$ $[\omega\uparrow n$, *ord-of-nat k*$]$ *2*
⟨*proof*⟩

end

3 An ordinal partition theorem by Jean A. Larson

Jean A. Larson, A short proof of a partition theorem for the ordinal ω^ω. *Annals of Mathematical Logic*, 6:129–145, 1973.

theory *Omega-Omega*
imports *HOL−Library.Product-Lexorder Erdos-Milner*

begin

abbreviation *list-of* ≡ *sorted-list-of-set*

3.1 Cantor normal form for ordinals below $\omega \uparrow \omega$

Unlike *Cantor-sum*, there is no list of ordinal exponents, which are instead taken as consecutive. We obtain an order-isomorphism between $\omega \uparrow \omega$ and increasing lists of natural numbers (ordered lexicographically).

fun *omega-sum-aux* **where**
Nil: *omega-sum-aux 0 - = 0*
| *Suc*: *omega-sum-aux* (*Suc n*) [] = *0*
| *Cons*: *omega-sum-aux* (*Suc n*) (*m#ms*) = $(\omega\uparrow n)$ ∗ (*ord-of-nat m*) + *omega-sum-aux n ms*

abbreviation *omega-sum* **where** *omega-sum ms* ≡ *omega-sum-aux* (*length ms*) *ms*

A normal expansion has no leading zeroes

inductive *normal*:: *nat list* ⇒ *bool* **where**
normal-Nil[*iff*]: *normal* []
| *normal-Cons*: *m* > *0* ⟹ *normal* (*m#ms*)

inductive-simps *normal-Cons-iff* [*simp*]: *normal* (*m#ms*)

lemma *omega-sum-0-iff* [*simp*]: *normal ns* ⟹ *omega-sum ns* = *0* ⟷ *ns* = []
⟨*proof*⟩

lemma *Ord-omega-sum-aux* [*simp*]: *Ord* (*omega-sum-aux k ms*)

⟨*proof*⟩

lemma *Ord-omega-sum*: $Ord\ (omega\text{-}sum\ ms)$
⟨*proof*⟩

lemma *omega-sum-less-ωω* [*intro*]: $omega\text{-}sum\ ms < \omega\uparrow\omega$
⟨*proof*⟩

lemma *omega-sum-aux-less*: $omega\text{-}sum\text{-}aux\ k\ ms < \omega \uparrow k$
⟨*proof*⟩

lemma *omega-sum-less*: $omega\text{-}sum\ ms < \omega \uparrow (length\ ms)$
⟨*proof*⟩

lemma *omega-sum-ge*: $m \neq 0 \Longrightarrow \omega \uparrow (length\ ms) \leq omega\text{-}sum\ (m\#ms)$
⟨*proof*⟩

lemma *omega-sum-length-less*:
assumes $normal\ ns\ length\ ms < length\ ns$
shows $omega\text{-}sum\ ms < omega\text{-}sum\ ns$
⟨*proof*⟩

lemma *omega-sum-length-leD*:
assumes $omega\text{-}sum\ ms \leq omega\text{-}sum\ ns\ normal\ ms$
shows $length\ ms \leq length\ ns$
⟨*proof*⟩

lemma *omega-sum-less-eqlen-iff-cases* [*simp*]:
assumes $length\ ms = length\ ns$
shows $omega\text{-}sum\ (m\#ms) < omega\text{-}sum\ (n\#ns) \longleftrightarrow m<n \vee m=n \wedge omega\text{-}sum\ ms < omega\text{-}sum\ ns$
⟨*proof*⟩

lemma *omega-sum-less-iff-cases*:
assumes $m > 0\ n > 0$
shows $omega\text{-}sum\ (m\#ms) < omega\text{-}sum\ (n\#ns)$
$\longleftrightarrow length\ ms < length\ ns$
$\vee\ length\ ms = length\ ns \wedge m<n$
$\vee\ length\ ms = length\ ns \wedge m=n \wedge omega\text{-}sum\ ms < omega\text{-}sum\ ns$
⟨*proof*⟩

lemma *omega-sum-less-iff*:
$((length\ ms,\ omega\text{-}sum\ ms),\ (length\ ns,\ omega\text{-}sum\ ns)) \in less\text{-}than <\!*lex*\!> VWF$
$\longleftrightarrow (ms,ns) \in lenlex\ less\text{-}than$
⟨*proof*⟩

lemma *eq-omega-sum-less-iff*:
assumes $length\ ms = length\ ns$

shows $(omega\text{-}sum\ ms,\ omega\text{-}sum\ ns) \in VWF \longleftrightarrow (ms,ns) \in lenlex\ less\text{-}than$
$\langle proof \rangle$

lemma *eq-omega-sum-eq-iff*:
assumes $length\ ms = length\ ns$
shows $omega\text{-}sum\ ms = omega\text{-}sum\ ns \longleftrightarrow ms=ns$
$\langle proof \rangle$

lemma *inj-omega-sum*: $inj\text{-}on\ omega\text{-}sum\ \{l.\ length\ l = n\}$
$\langle proof \rangle$

lemma *Ex-omega-sum*: $\gamma \in elts\ (\omega\uparrow n) \Longrightarrow \exists\, ns.\ \gamma = omega\text{-}sum\ ns \wedge length\ ns = n$
$\langle proof \rangle$

lemma *omega-sum-drop* [*simp*]: $omega\text{-}sum\ (dropWhile\ (\lambda n.\ n=0)\ ns) = omega\text{-}sum\ ns$
$\langle proof \rangle$

lemma *normal-drop* [*simp*]: $normal\ (dropWhile\ (\lambda n.\ n=0)\ ns)$
$\langle proof \rangle$

lemma *omega-sum-ωω*:
assumes $\gamma \in elts\ (\omega\uparrow\omega)$
obtains *ns* **where** $\gamma = omega\text{-}sum\ ns\ normal\ ns$
$\langle proof \rangle$

definition $Cantor\text{-}\omega\omega :: V \Rightarrow nat\ list$
where $Cantor\text{-}\omega\omega \equiv \lambda x.\ SOME\ ns.\ x = omega\text{-}sum\ ns \wedge normal\ ns$

lemma
assumes $\gamma \in elts\ (\omega\uparrow\omega)$
shows *Cantor-ωω*: $omega\text{-}sum\ (Cantor\text{-}\omega\omega\ \gamma) = \gamma$
and *normal-Cantor-ωω*: $normal\ (Cantor\text{-}\omega\omega\ \gamma)$
$\langle proof \rangle$

3.2 Larson's set $W(n)$

definition $WW :: nat\ list\ set$
where $WW \equiv \{l.\ strict\text{-}sorted\ l\}$

fun $into\text{-}WW :: nat \Rightarrow nat\ list \Rightarrow nat\ list$ **where**
$into\text{-}WW\ k\ [] = []$
$|\ into\text{-}WW\ k\ (n\#ns) = (k+n)\ \#\ into\text{-}WW\ (Suc\ (k+n))\ ns$

fun $from\text{-}WW :: nat \Rightarrow nat\ list \Rightarrow nat\ list$ **where**
$from\text{-}WW\ k\ [] = []$
$|\ from\text{-}WW\ k\ (n\#ns) = (n - k)\ \#\ from\text{-}WW\ (Suc\ n)\ ns$

lemma *from-into-WW* [*simp*]: *from-WW k* (*into-WW k ns*) = *ns*
⟨*proof*⟩

lemma *inj-into-WW*: *inj* (*into-WW k*)
⟨*proof*⟩

lemma *into-from-WW-aux*:
⟦*strict-sorted ns*; ∀ *n*∈*list.set ns*. *k* ≤ *n*⟧ ⟹ *into-WW k* (*from-WW k ns*) = *ns*
⟨*proof*⟩

lemma *into-from-WW* [*simp*]: *strict-sorted ns* ⟹ *into-WW 0* (*from-WW 0 ns*) = *ns*
⟨*proof*⟩

lemma *into-WW-imp-ge*: *y* ∈ *List.set* (*into-WW x ns*) ⟹ *x* ≤ *y*
⟨*proof*⟩

lemma *strict-sorted-into-WW*: *strict-sorted* (*into-WW x ns*)
⟨*proof*⟩

lemma *length-into-WW*: *length* (*into-WW x ns*) = *length ns*
⟨*proof*⟩

lemma *WW-eq-range-into*: *WW* = *range* (*into-WW 0*)
⟨*proof*⟩

lemma *into-WW-lenlex-iff*: (*into-WW k ms*, *into-WW k ns*) ∈ *lenlex less-than* ⟷ (*ms*, *ns*) ∈ *lenlex less-than*
⟨*proof*⟩

lemma *wf-llt* [*simp*]: *wf* (*lenlex less-than*) **and** *trans-llt* [*simp*]: *trans* (*lenlex less-than*)
⟨*proof*⟩

lemma *total-llt* [*simp*]: *total-on A* (*lenlex less-than*)
⟨*proof*⟩

lemma *omega-sum-1-less*:
assumes (*ms*,*ns*) ∈ *lenlex less-than* **shows** *omega-sum* (*1*#*ms*) < *omega-sum* (*1*#*ns*)
⟨*proof*⟩

lemma *ordertype-WW-1*: *ordertype WW* (*lenlex less-than*) ≤ *ordertype UNIV* (*lenlex less-than*)
⟨*proof*⟩

lemma *ordertype-WW-2*: *ordertype UNIV* (*lenlex less-than*) ≤ $\omega\uparrow\omega$
⟨*proof*⟩

lemma *ordertype-WW-3*: $\omega\uparrow\omega$ ≤ *ordertype WW* (*lenlex less-than*)

⟨*proof*⟩

lemma *ordertype-WW*: *ordertype WW* (*lenlex less-than*) = $\omega\uparrow\omega$
and *ordertype-UNIV-ωω*: *ordertype UNIV* (*lenlex less-than*) = $\omega\uparrow\omega$
⟨*proof*⟩

lemma *ordertype-ωω*:
fixes F :: *nat* ⇒ *nat list set*
assumes $\bigwedge j$::*nat*. *ordertype* (F j) (*lenlex less-than*) = $\omega\uparrow j$
shows *ordertype* ($\bigcup j$. F j) (*lenlex less-than*) = $\omega\uparrow\omega$
⟨*proof*⟩

definition *WW-seg* :: *nat* ⇒ *nat list set*
where *WW-seg* n ≡ {l ∈ *WW*. *length* l = n}

lemma *WW-seg-subset-WW*: *WW-seg* n ⊆ *WW*
⟨*proof*⟩

lemma *WW-eq-UN-WW-seg*: *WW* = ($\bigcup$ n. *WW-seg* n)
⟨*proof*⟩

lemma *ordertype-list-seg*: *ordertype* {l. *length* l = n} (*lenlex less-than*) = $\omega\uparrow n$
⟨*proof*⟩

lemma *ordertype-WW-seg*: *ordertype* (*WW-seg* n) (*lenlex less-than*) = $\omega\uparrow n$
(**is** *ordertype ?W ?R* = $\omega\uparrow n$)
⟨*proof*⟩

3.3 Definitions required for the lemmas

3.3.1 Larson's "<"-relation on ordered lists

instantiation *list* :: (*ord*)*ord*
begin

definition $xs < ys \equiv xs \neq [] \wedge ys \neq [] \longrightarrow$ *last* $xs <$ *hd* ys **for** xs ys :: *'a list*
definition $xs \le ys \equiv xs < ys \vee xs = ys$ **for** xs ys :: *'a list*

instance
⟨*proof*⟩

end

lemma *less-Nil* [*simp*]: $xs < []$ $[] < xs$
⟨*proof*⟩

lemma *less-sets-imp-list-less*:

assumes *list.set xs* $\ll$ *list.set ys*
shows *xs* $<$ *ys*
⟨*proof*⟩

lemma *less-sets-imp-sorted-list-of-set*:
assumes $A \ll B$ *finite A finite B*
shows *list-of A* $<$ *list-of B*
⟨*proof*⟩

lemma *sorted-list-of-set-imp-less-sets*:
assumes *xs* $<$ *ys sorted xs sorted ys*
shows *list.set xs* $\ll$ *list.set ys*
⟨*proof*⟩

lemma *less-list-iff-less-sets*:
assumes *sorted xs sorted ys*
shows *xs* $<$ *ys* $\longleftrightarrow$ *list.set xs* $\ll$ *list.set ys*
⟨*proof*⟩

lemma *strict-sorted-append-iff*:
strict-sorted (*xs* @ *ys*) $\longleftrightarrow$ *xs* $<$ *ys* $\wedge$ *strict-sorted xs* $\wedge$ *strict-sorted ys*
⟨*proof*⟩

lemma *singleton-less-list-iff*: *sorted xs* $\Longrightarrow$ $[n] <$ *xs* $\longleftrightarrow$ $\{..n\} \cap$ *list.set xs* $= \{\}$
⟨*proof*⟩

lemma *less-hd-imp-less*: *xs* $<$ [*hd ys*] $\Longrightarrow$ *xs* $<$ *ys*
⟨*proof*⟩

lemma *strict-sorted-concat-I*:
assumes $\bigwedge x.\ x \in$ *list.set xs* $\Longrightarrow$ *strict-sorted x*
$\bigwedge n.$ *Suc n* $<$ *length xs* $\Longrightarrow$ *xs*!*n* $<$ *xs*!*Suc n*
xs $\in$ *lists* $(- \{[]\})$
shows *strict-sorted* (*concat xs*)
⟨*proof*⟩

3.4 Nash Williams for lists

3.4.1 Thin sets of lists

inductive *initial-segment* :: *'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *bool*
where *initial-segment xs* (*xs*@*ys*)

definition *thin* :: *'a list set* $\Rightarrow$ *bool*
where *thin A* $\equiv \neg$ $(\exists\, x\ y.\ x \in A \wedge y \in A \wedge x \neq y \wedge$ *initial-segment x y*$)$

lemma *initial-segment-ne*:
assumes *initial-segment xs ys xs* $\neq []$
shows *ys* $\neq [] \wedge$ *hd ys* $=$ *hd xs*
⟨*proof*⟩

lemma *take-initial-segment*:
  assumes *initial-segment xs ys* $k \leq$ *length xs*
  shows *take k xs* = *take k ys*
  ⟨*proof*⟩

lemma *initial-segment-length-eq*:
  assumes *initial-segment xs ys length xs* = *length ys*
  shows *xs* = *ys*
  ⟨*proof*⟩

lemma *initial-segment-Nil* [*simp*]: *initial-segment* [] *ys*
  ⟨*proof*⟩

lemma *initial-segment-Cons* [*simp*]: *initial-segment* (*x#xs*) (*y#ys*) $\longleftrightarrow$ *x=y* $\wedge$ *initial-segment xs ys*
  ⟨*proof*⟩

lemma *init-segment-iff-initial-segment*:
  assumes *strict-sorted xs strict-sorted ys*
  shows *init-segment* (*list.set xs*) (*list.set ys*) $\longleftrightarrow$ *initial-segment xs ys* (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

theorem *Nash-Williams-WW*:
  fixes *h* :: *nat list* $\Rightarrow$ *nat*
  assumes *infinite M* **and** *h*: *h* ‘ {*l* $\in$ *A*. *List.set l* $\subseteq$ *M*} $\subseteq$ {..<*2*} **and** *thin A A* $\subseteq$ *WW*
  obtains *i N* **where** *i* < *2 infinite N N* $\subseteq$ *M h* ‘ {*l* $\in$ *A*. *List.set l* $\subseteq$ *N*} $\subseteq$ {*i*}
⟨*proof*⟩

3.5 Specialised functions on lists

lemma *mem-lists-non-Nil*: *xss* $\in$ *lists* (− {[]}) $\longleftrightarrow$ ($\forall$ *x* $\in$ *list.set xss*. *x* $\neq$ [])
  ⟨*proof*⟩

fun *acc-lengths* :: *nat* $\Rightarrow$ *'a list list* $\Rightarrow$ *nat list*
  where *acc-lengths acc* [] = []
    | *acc-lengths acc* (*l#ls*) = (*acc* + *length l*) # *acc-lengths* (*acc* + *length l*) *ls*

lemma *length-acc-lengths* [*simp*]: *length* (*acc-lengths acc ls*) = *length ls*
  ⟨*proof*⟩

lemma *acc-lengths-eq-Nil-iff* [*simp*]: *acc-lengths acc ls* = [] $\longleftrightarrow$ *ls* = []
  ⟨*proof*⟩

lemma *set-acc-lengths*:
  assumes *ls* $\in$ *lists* (− {[]}) **shows** *list.set* (*acc-lengths acc ls*) $\subseteq$ {*acc*<..}
  ⟨*proof*⟩

Useful because *acc-lengths.simps* will sometimes be deleted from the simpset.

lemma *hd-acc-lengths* [*simp*]: *hd* (*acc-lengths acc* (*l#ls*)) = *acc* + *length l*
⟨*proof*⟩

lemma *last-acc-lengths* [*simp*]:
ls ≠ [] ⟹ *last* (*acc-lengths acc ls*) = *acc* + *sum-list* (*map length ls*)
⟨*proof*⟩

lemma *nth-acc-lengths* [*simp*]:
⟦*ls* ≠ []; *k* < *length ls*⟧ ⟹ *acc-lengths acc ls* ! *k* = *acc* + *sum-list* (*map length* (*take* (*Suc k*) *ls*))
⟨*proof*⟩

lemma *acc-lengths-plus*: *acc-lengths* (*m*+*n*) *as* = *map* ((+)*m*) (*acc-lengths n as*)
⟨*proof*⟩

lemma *acc-lengths-shift*: *NO-MATCH 0 acc* ⟹ *acc-lengths acc as* = *map* ((+)*acc*) (*acc-lengths 0 as*)
⟨*proof*⟩

lemma *length-concat-acc-lengths*:
ls ≠ [] ⟹ *k* + *length* (*concat ls*) ∈ *list.set* (*acc-lengths k ls*)
⟨*proof*⟩

lemma *strict-sorted-acc-lengths*:
assumes *ls* ∈ *lists* (− {[]}) **shows** *strict-sorted* (*acc-lengths acc ls*)
⟨*proof*⟩

lemma *acc-lengths-append*:
acc-lengths acc (*xs* @ *ys*)
= *acc-lengths acc xs* @ *acc-lengths* (*acc* + *sum-list* (*map length xs*)) *ys*
⟨*proof*⟩

lemma *length-concat-ge*:
assumes *as* ∈ *lists* (− {[]})
shows *length* (*concat as*) ≥ *length as*
⟨*proof*⟩

fun *interact* :: *'a list list* ⇒ *'a list list* ⇒ *'a list*
where
interact [] *ys* = *concat ys*
| *interact xs* [] = *concat xs*
| *interact* (*x#xs*) (*y#ys*) = *x* @ *y* @ *interact xs ys*

lemma (**in** *monoid-add*) *length-interact*:
length (*interact xs ys*) = *sum-list* (*map length xs*) + *sum-list* (*map length ys*)

$\langle proof \rangle$

lemma *length-interact-ge*:
assumes $xs \in lists\ (- \{[]\})$ $ys \in lists\ (- \{[]\})$
shows $length\ (interact\ xs\ ys) \geq length\ xs + length\ ys$
$\langle proof \rangle$

lemma *set-interact* [*simp*]:
shows $list.set\ (interact\ xs\ ys) = list.set\ (concat\ xs) \cup list.set\ (concat\ ys)$
$\langle proof \rangle$

lemma *interact-eq-Nil-iff* [*simp*]:
assumes $xs \in lists\ (- \{[]\})$ $ys \in lists\ (- \{[]\})$
shows $interact\ xs\ ys = [] \longleftrightarrow xs=[] \wedge ys=[]$
$\langle proof \rangle$

lemma *interact-sing* [*simp*]: $interact\ [x]\ ys = x\ @\ concat\ ys$
$\langle proof \rangle$

lemma *hd-interact*: $[\![xs \neq [];\ hd\ xs \neq []]\!] \Longrightarrow hd\ (interact\ xs\ ys) = hd\ (hd\ xs)$
$\langle proof \rangle$

lemma *acc-lengths-concat-injective*:
assumes $concat\ as' = concat\ as$ $acc\text{-}lengths\ n\ as' = acc\text{-}lengths\ n\ as$
shows $as' = as$
$\langle proof \rangle$

lemma *acc-lengths-interact-injective*:
assumes $interact\ as'\ bs' = interact\ as\ bs$ $acc\text{-}lengths\ a\ as' = acc\text{-}lengths\ a\ as$ $acc\text{-}lengths\ b\ bs' = acc\text{-}lengths\ b\ bs$
shows $as' = as \wedge bs' = bs$
$\langle proof \rangle$

lemma *strict-sorted-interact-I*:
assumes $length\ ys \leq length\ xs$ $length\ xs \leq Suc\ (length\ ys)$
$\bigwedge x.\ x \in list.set\ xs \Longrightarrow strict\text{-}sorted\ x$
$\bigwedge y.\ y \in list.set\ ys \Longrightarrow strict\text{-}sorted\ y$
$\bigwedge n.\ n < length\ ys \Longrightarrow xs!n < ys!n$
$\bigwedge n.\ Suc\ n < length\ xs \Longrightarrow ys!n < xs!Suc\ n$
assumes $xs \in lists\ (- \{[]\})$ $ys \in lists\ (- \{[]\})$
shows $strict\text{-}sorted\ (interact\ xs\ ys)$
$\langle proof \rangle$

3.6 Forms and interactions

3.6.1 Forms

inductive *Form-Body* :: $[nat,\ nat,\ nat\ list,\ nat\ list,\ nat\ list] \Rightarrow bool$
where $Form\text{-}Body\ ka\ kb\ xs\ ys\ zs$

if *length xs* < *length ys xs* = *concat* (*a*#*as*) *ys* = *concat* (*b*#*bs*)
a#*as* ∈ *lists* (− {[]}) *b*#*bs* ∈ *lists* (− {[]})
length (*a*#*as*) = *ka length* (*b*#*bs*) = *kb*
c = *acc-lengths 0* (*a*#*as*)
d = *acc-lengths 0* (*b*#*bs*)
zs = *concat* [*c*, *a*, *d*, *b*] @ *interact as bs*
strict-sorted zs

inductive *Form* :: [*nat*, *nat list set*] ⇒ *bool*
where *Form 0* {*xs*,*ys*} **if** *length xs* = *length ys xs* ≠ *ys*
| *Form* (2∗*k*−1) {*xs*,*ys*} **if** *Form-Body k k xs ys zs k* > *0*
| *Form* (2∗*k*) {*xs*,*ys*} **if** *Form-Body* (*Suc k*) *k xs ys zs k* > *0*

inductive-cases *Form-0-cases-raw*: *Form 0 u*

lemma *Form-elim-upair*:
assumes *Form l U*
obtains *xs ys* **where** *xs* ≠ *ys U* = {*xs*,*ys*} *length xs* ≤ *length ys*
⟨*proof*⟩

lemma assumes *Form-Body ka kb xs ys zs*
shows *Form-Body-WW*: *zs* ∈ *WW*
and *Form-Body-nonempty*: *length zs* > *0*
and *Form-Body-length*: *length xs* < *length ys*
⟨*proof*⟩

lemma *form-cases*:
fixes *l*::*nat*
obtains (*zero*) *l* = *0* | (*nz*) *ka kb* **where** *l* = *ka*+*kb* − *1 0* < *kb kb* ≤ *ka ka* ≤ *Suc kb*
⟨*proof*⟩

3.6.2 Interactions

lemma *interact*:
assumes *Form l U l*>*0*
obtains *ka kb xs ys zs* **where** *l* = *ka*+*kb* − *1 U* = {*xs*,*ys*} *Form-Body ka kb xs ys zs 0* < *kb kb* ≤ *ka ka* ≤ *Suc kb*
⟨*proof*⟩

definition *inter-scheme* :: *nat* ⇒ *nat list set* ⇒ *nat list*
where *inter-scheme l U* ≡
SOME zs. ∃ *k xs ys*. *U* = {*xs*,*ys*} ∧
(*l* = 2∗*k*−1 ∧ *Form-Body k k xs ys zs* ∨ *l* = 2∗*k* ∧ *Form-Body* (*Suc k*) *k xs ys zs*)

lemma *inter-scheme*:
assumes *Form l U l>0*
obtains *ka kb xs ys* **where** $l = ka+kb - 1$ $U = \{xs,ys\}$ *Form-Body ka kb xs ys* (*inter-scheme l U*) $0 < kb$ $kb \leq ka$ $ka \leq Suc\ kb$
⟨*proof*⟩

lemma *inter-scheme-strict-sorted*:
assumes *Form l U l>0*
shows *strict-sorted* (*inter-scheme l U*)
⟨*proof*⟩

lemma *inter-scheme-simple*:
assumes *Form l U l>0*
shows *inter-scheme l U* $\in$ *WW* $\wedge$ *length* (*inter-scheme l U*) > 0
⟨*proof*⟩

3.6.3 Injectivity of interactions

proposition *inter-scheme-injective*:
assumes *Form l U Form l U′* $l > 0$ **and** *eq*: *inter-scheme l U′* = *inter-scheme l U*
shows $U' = U$
⟨*proof*⟩

lemma *strict-sorted-interact-imp-concat*:
strict-sorted (*interact as bs*) $\Longrightarrow$ *strict-sorted* (*concat as*) $\wedge$ *strict-sorted* (*concat bs*)
⟨*proof*⟩

lemma *strict-sorted-interact-hd*:
⟦*strict-sorted* (*interact cs ds*); $cs \neq []$; $ds \neq []$; $hd\ cs \neq []$; $hd\ ds \neq []$⟧
$\Longrightarrow hd\ (hd\ cs) < hd\ (hd\ ds)$
⟨*proof*⟩

the lengths of the two lists can differ by one

proposition *interaction-scheme-unique-aux*:
assumes *concat as* = *concat as′* **and** *ys′*: *concat bs* = *concat bs′*
and *as* $\in$ *lists* $(- \{[]\})$ *bs* $\in$ *lists* $(- \{[]\})$
and *strict-sorted* (*interact as bs*)
and *length bs* $\leq$ *length as length as* $\leq$ *Suc* (*length bs*)
and *as′* $\in$ *lists* $(- \{[]\})$ *bs′* $\in$ *lists* $(- \{[]\})$
and *strict-sorted* (*interact as′ bs′*)
and *length bs′* $\leq$ *length as′ length as′* $\leq$ *Suc* (*length bs′*)
and *length as* = *length as′ length bs* = *length bs′*
shows *as* = *as′* $\wedge$ *bs* = *bs′*
⟨*proof*⟩

proposition *Form-Body-unique*:
assumes *Form-Body ka kb xs ys zs Form-Body ka kb xs ys zs'* **and** $kb \leq ka$ $ka \leq$ *Suc kb*
shows $zs' = zs$
⟨*proof*⟩

lemma *Form-Body-imp-inter-scheme*:
assumes *FB*: *Form-Body ka kb xs ys zs* **and** $0 < kb$ $kb \leq ka$ $ka \leq$ *Suc kb*
shows $zs =$ *inter-scheme* $((ka+kb) - Suc\ 0)$ $\{xs,ys\}$
⟨*proof*⟩

3.7 For Lemma 3.8 AND PROBABLY 3.7

definition *grab* :: $nat\ set \Rightarrow nat \Rightarrow nat\ set \times nat\ set$
where $grab\ N\ n \equiv (N \cap enumerate\ N\ `\ \{..<n\},\ N \cap \{enumerate\ N\ n..\})$

lemma *grab-0* [*simp*]: $grab\ N\ 0 = (\{\},\ N)$
⟨*proof*⟩

lemma *less-sets-grab*:
$infinite\ N \Longrightarrow fst\ (grab\ N\ n) \ll snd\ (grab\ N\ n)$
⟨*proof*⟩

lemma *finite-grab* [*iff*]: $finite\ (fst\ (grab\ N\ n))$
⟨*proof*⟩

lemma *card-grab* [*simp*]:
assumes *infinite N* **shows** $card\ (fst\ (grab\ N\ n)) = n$
⟨*proof*⟩

lemma *fst-grab-subset*: $fst\ (grab\ N\ n) \subseteq N$
⟨*proof*⟩

lemma *snd-grab-subset*: $snd\ (grab\ N\ n) \subseteq N$
⟨*proof*⟩

lemma *grab-Un-eq*:
assumes *infinite N* **shows** $fst\ (grab\ N\ n) \cup snd\ (grab\ N\ n) = N$
⟨*proof*⟩

lemma *finite-grab-iff* [*simp*]: $finite\ (snd\ (grab\ N\ n)) \longleftrightarrow finite\ N$
⟨*proof*⟩

lemma *grab-eqD*:
$[\![grab\ N\ n = (A,M);\ infinite\ N]\!]$
$\Longrightarrow A \ll M \wedge finite\ A \wedge card\ A = n \wedge infinite\ M \wedge A \subseteq N \wedge M \subseteq N$
⟨*proof*⟩

lemma *less-sets-fst-grab*: $A \ll N \Longrightarrow A \ll fst\ (grab\ N\ n)$
⟨*proof*⟩

Possibly redundant, given *grab*

definition *nxt* **where** $nxt \equiv \lambda N.\ \lambda n{::}nat.\ N \cap \{n{<}..\}$

lemma *infinite-nxtN*: $infinite\ N \Longrightarrow infinite\ (nxt\ N\ n)$
⟨*proof*⟩

lemma *nxt-subset*: $nxt\ N\ n \subseteq N$
⟨*proof*⟩

lemma *nxt-subset-greaterThan*: $m \le n \Longrightarrow nxt\ N\ n \subseteq \{m{<}..\}$
⟨*proof*⟩

lemma *nxt-subset-atLeast*: $m \le n \Longrightarrow nxt\ N\ n \subseteq \{m..\}$
⟨*proof*⟩

lemma *enum-nxt-ge*: $infinite\ N \Longrightarrow a \le enum\ (nxt\ N\ a)\ n$
⟨*proof*⟩

lemma *inj-enum-nxt*: $infinite\ N \Longrightarrow inj\text{-}on\ (enum\ (nxt\ N\ a))\ A$
⟨*proof*⟩

3.8 Larson's Lemma 3.11

Again from Jean A. Larson, A short proof of a partition theorem for the ordinal ω^ω. *Annals of Mathematical Logic*, 6:129–145, 1973.

lemma *lemma-3-11*:
assumes $l > 0$
shows $thin\ (inter\text{-}scheme\ l\ `\ \{U.\ Form\ l\ U\})$
⟨*proof*⟩

3.9 Larson's Lemma 3.6

proposition *lemma-3-6*:
fixes $g :: nat\ list\ set \Rightarrow nat$
assumes *g*: $g \in [WW]^2 \rightarrow \{0,1\}$
obtains *N j* **where** *infinite N*
and $\bigwedge k\ u.\ [\![k > 0;\ u \in [WW]^2;\ Form\ k\ u;\ [enum\ N\ k] < inter\text{-}scheme\ k\ u;\ List.set\ (inter\text{-}scheme\ k\ u) \subseteq N]\!] \Longrightarrow g\ u = j\ k$
⟨*proof*⟩

3.10 Larson's Lemma 3.7

3.10.1 Preliminaries

Analogous to *ordered-nsets-2-eq*, but without type classes

lemma *total-order-nsets-2-eq*:
  assumes *tot*: *total-on A r* **and** *irr*: *irrefl r*
  shows *nsets A 2* $= \{\{x,y\} \mid x\ y.\ x \in A \wedge y \in A \wedge (x,y) \in r\}$
    (**is** - = *?rhs*)
⟨*proof*⟩

lemma *lenlex-nsets-2-eq*: *nsets A 2* $= \{\{x,y\} \mid x\ y.\ x \in A \wedge y \in A \wedge (x,y) \in$ *lenlex less-than*$\}$
  ⟨*proof*⟩

lemma *sum-sorted-list-of-set-map*: *finite I* $\Longrightarrow$ *sum-list* (*map f* (*list-of I*)) = *sum f I*
⟨*proof*⟩

lemma *sorted-list-of-set-UN-eq-concat*:
  assumes *I*: *strict-mono-sets I f finite I* **and** *fin*: $\bigwedge i.$ *finite* (*f i*)
  shows *list-of* $(\bigcup i \in I.\ f\ i)$ = *concat* (*map* (*list-of* $\circ$ *f*) (*list-of I*))
  ⟨*proof*⟩

3.10.2 Lemma 3.7 of Jean A. Larson, ibid.

proposition *lemma-3-7*:
  assumes *infinite N* $l > 0$
  obtains *M* **where** $M \in [WW]^m$
    $\bigwedge U.\ U \in [M]^2 \Longrightarrow$ *Form l U* $\wedge$ *List.set* (*inter-scheme l U*) $\subseteq N$
⟨*proof*⟩

3.11 Larson's Lemma 3.8

3.11.1 Primitives needed for the inductive construction of *b*

definition *IJ* **where** *IJ* $\equiv \lambda k.$ *Sigma* $\{..k\}$ $(\lambda j{::}nat.\ \{..{<}j\})$

lemma *IJ-iff*: $u \in IJ\ k \longleftrightarrow (\exists j\ i.\ u = (j,i) \wedge i{<}j \wedge j{\le}k)$
  ⟨*proof*⟩

lemma *finite-IJ*: *finite* (*IJ k*)
  ⟨*proof*⟩

fun *prev* **where**
  prev 0 0 = None
| *prev* (*Suc 0*) *0 = None*
| *prev* (*Suc j*) *0 = Some* (*j*, $j -$ *Suc 0*)
| *prev j* (*Suc i*) = *Some* (*j*,*i*)

lemma *prev-eq-None-iff*: *prev j i = None* $\longleftrightarrow j \le$ *Suc 0* $\wedge\ i = 0$
  ⟨*proof*⟩

lemma *prev-pair-less*:

$prev\ j\ i = Some\ ji' \Longrightarrow (ji', (j,i)) \in pair\text{-}less$
⟨*proof*⟩

lemma *prev-Some-less*: $[\![prev\ j\ i = Some\ (j',i');\ i \le j]\!] \Longrightarrow i' < j'$
⟨*proof*⟩

lemma *prev-maximal*:
$[\![prev\ j\ i = Some\ (j',i');\ (ji'', (j,i)) \in pair\text{-}less;\ ji'' \in IJ\ k]\!]$
$\Longrightarrow (ji'', (j',i')) \in pair\text{-}less \lor ji'' = (j',i')$
⟨*proof*⟩

lemma *pair-less-prev*:
assumes $(u, (j,i)) \in pair\text{-}less\ u \in IJ\ k$
shows $prev\ j\ i = Some\ u \lor (\exists x.\ (u, x) \in pair\text{-}less \land prev\ j\ i = Some\ x)$
⟨*proof*⟩

3.11.2 Special primitives for the ordertype proof

definition *USigma* :: $'a\ set\ set \Rightarrow ('a\ set \Rightarrow 'a\ set) \Rightarrow 'a\ set\ set$
where $USigma\ \mathcal{A}\ B \equiv \bigcup X{\in}\mathcal{A}.\ \bigcup y \in B\ X.\ \{insert\ y\ X\}$

definition *usplit*
where $usplit\ f\ A \equiv f\ (A - \{Max\ A\})\ (Max\ A)$

lemma *USigma-empty* [*simp*]: $USigma\ \{\}\ B = \{\}$
⟨*proof*⟩

lemma *USigma-iff*:
assumes $\bigwedge I\ j.\ I \in \mathcal{I} \Longrightarrow I \ll J\ I \land finite\ I$
shows $x \in USigma\ \mathcal{I}\ J \longleftrightarrow usplit\ (\lambda I\ j.\ I \in \mathcal{I} \land j \in J\ I \land x = insert\ j\ I)\ x$
⟨*proof*⟩

proposition *ordertype-append-image-IJ*:
assumes *lenB* [*simp*]: $\bigwedge i\ j.\ i \in \mathcal{I} \Longrightarrow j \in J\ i \Longrightarrow length\ (B\ j) = c$
and *AB*: $\bigwedge i\ j.\ i \in \mathcal{I} \Longrightarrow j \in J\ i \Longrightarrow A\ i < B\ j$
and *IJ*: $\bigwedge i.\ i \in \mathcal{I} \Longrightarrow i \ll J\ i \land finite\ i$
and β: $\bigwedge i.\ i \in \mathcal{I} \Longrightarrow ordertype\ (B\ `\ J\ i)\ (lenlex\ less\text{-}than) = \beta$
and *A*: $inj\text{-}on\ A\ \mathcal{I}$
shows $ordertype\ (usplit\ (\lambda i\ j.\ A\ i\ @\ B\ j)\ `\ USigma\ \mathcal{I}\ J)\ (lenlex\ less\text{-}than)$
$= \beta * ordertype\ (A\ `\ \mathcal{I})\ (lenlex\ less\text{-}than)$
(**is** $ordertype\ ?AB\ ?R = - * ?\alpha$)
⟨*proof*⟩

3.11.3 The final part of 3.8, where two sequences are merged

inductive *merge* :: $[nat\ list\ list, nat\ list\ list, nat\ list\ list, nat\ list\ list] \Rightarrow bool$
where *NullNull*: $merge\ []\ []\ []\ []$
| *Null*: $as \neq [] \Longrightarrow merge\ as\ []\ [concat\ as]\ []$
| *App*: $[\![as1 \neq [];\ bs1 \neq [];$

$concat\ as1 < concat\ bs1;\ concat\ bs1 < concat\ as2;\ merge\ as2\ bs2\ as\ bs\rrbracket$
$\Longrightarrow merge\ (as1@as2)\ (bs1@bs2)\ (concat\ as1\ \#\ as)\ (concat\ bs1\ \#\ bs)$

inductive-simps *Null1* [*simp*]: *merge* [] *bs us vs*
inductive-simps *Null2* [*simp*]: *merge as* [] *us vs*

lemma *merge-single*:
$\llbracket concat\ as < concat\ bs;\ concat\ as \neq [];\ concat\ bs \neq []\rrbracket \Longrightarrow merge\ as\ bs\ [concat\ as]\ [concat\ bs]$
$\langle proof\rangle$

lemma *merge-length1-nonempty*:
assumes $merge\ as\ bs\ us\ vs\ as \in lists\ (- \{[]\})$
shows $us \in lists\ (- \{[]\})$
$\langle proof\rangle$

lemma *merge-length2-nonempty*:
assumes $merge\ as\ bs\ us\ vs\ bs \in lists\ (- \{[]\})$
shows $vs \in lists\ (- \{[]\})$
$\langle proof\rangle$

lemma *merge-length1-gt-0*:
assumes $merge\ as\ bs\ us\ vs\ as \neq []$
shows $length\ us > 0$
$\langle proof\rangle$

lemma *merge-length-le*:
assumes *merge as bs us vs*
shows $length\ vs \leq length\ us$
$\langle proof\rangle$

lemma *merge-length-le-Suc*:
assumes *merge as bs us vs*
shows $length\ us \leq Suc\ (length\ vs)$
$\langle proof\rangle$

lemma *merge-length-less2*:
assumes *merge as bs us vs*
shows $length\ vs \leq length\ as$
$\langle proof\rangle$

lemma *merge-preserves*:
assumes *merge as bs us vs*
shows $concat\ as = concat\ us \wedge concat\ bs = concat\ vs$
$\langle proof\rangle$

lemma *merge-interact*:
assumes *merge as bs us vs strict-sorted* (*concat as*) *strict-sorted* (*concat bs*)

$bs \in lists\ (- \{[]\})$
shows $strict\text{-}sorted\ (interact\ us\ vs)$
$\langle proof \rangle$

lemma *acc-lengths-merge1*:
assumes $merge\ as\ bs\ us\ vs$
shows $list.set\ (acc\text{-}lengths\ k\ us) \subseteq list.set\ (acc\text{-}lengths\ k\ as)$
$\langle proof \rangle$

lemma *acc-lengths-merge2*:
assumes $merge\ as\ bs\ us\ vs$
shows $list.set\ (acc\text{-}lengths\ k\ vs) \subseteq list.set\ (acc\text{-}lengths\ k\ bs)$
$\langle proof \rangle$

lemma *length-hd-le-concat*:
assumes $as \neq []$ **shows** $length\ (hd\ as) \leq length\ (concat\ as)$
$\langle proof \rangle$

lemma *length-hd-merge2*:
assumes $merge\ as\ bs\ us\ vs$
shows $length\ (hd\ bs) \leq length\ (hd\ vs)$
$\langle proof \rangle$

lemma *merge-less-sets-hd*:
assumes $merge\ as\ bs\ us\ vs\ strict\text{-}sorted\ (concat\ as)\ strict\text{-}sorted\ (concat\ bs)\ bs \in lists\ (- \{[]\})$
shows $list.set\ (hd\ us) \ll list.set\ (concat\ vs)$
$\langle proof \rangle$

lemma *set-takeWhile*:
assumes $strict\text{-}sorted\ (concat\ as)\ as \in lists\ (- \{[]\})$
shows $list.set\ (takeWhile\ (\lambda x.\ x < y)\ as) = \{x \in list.set\ as.\ x < y\}$
$\langle proof \rangle$

proposition *merge-exists*:
assumes $strict\text{-}sorted\ (concat\ as)\ strict\text{-}sorted\ (concat\ bs)$
$as \in lists\ (- \{[]\})\ bs \in lists\ (- \{[]\})$
$hd\ as < hd\ bs\ as \neq []\ bs \neq []$
and *disj*: $\bigwedge a\ b.\ [\![a \in list.set\ as;\ b \in list.set\ bs]\!] \Longrightarrow a{<}b \lor b{<}a$
shows $\exists\, us\ vs.\ merge\ as\ bs\ us\ vs$
$\langle proof \rangle$

3.11.4 Actual proof of Larson's Lemma 3.8

proposition *lemma-3-8*:
assumes *infinite N*
obtains X **where** $X \subseteq WW\ ordertype\ X\ (lenlex\ less\text{-}than) = \omega{\uparrow}\omega$
$\bigwedge u.\ u \in [X]^2 \Longrightarrow$

$\exists l.$ *Form l u* $\wedge$ ($l > 0 \longrightarrow$ [*enum N l*] $<$ *inter-scheme l u* $\wedge$ *List.set* (*inter-scheme l u*) $\subseteq N$)
⟨*proof*⟩

3.12 The main partition theorem for $\omega \uparrow \omega$

definition *iso-ll* **where** *iso-ll A B* $\equiv$ *iso* (*lenlex less-than* $\cap$ ($A \times A$)) (*lenlex less-than* $\cap$ ($B \times B$))

corollary *ordertype-eq-ordertype-iso-ll*:
assumes *Field* (*Restr* (*lenlex less-than*) *A*) = *A* *Field* (*Restr* (*lenlex less-than*) *B*) = *B*
shows (*ordertype A* (*lenlex less-than*) = *ordertype B* (*lenlex less-than*))
$\longleftrightarrow$ ($\exists f.$ *iso-ll A B f*)
⟨*proof*⟩

theorem *partition-ωω-aux*:
assumes $\alpha \in$ *elts* ω
shows *partn-lst* (*lenlex less-than*) *WW* [$\omega\uparrow\omega$,α] *2* (**is** *partn-lst ?R WW* [$\omega\uparrow\omega$,α] *2*)
⟨*proof*⟩

Theorem 3.1 of Jean A. Larson, ibid.

theorem *partition-ωω*: $\alpha \in$ *elts* $\omega \Longrightarrow$ *partn-lst-VWF* ($\omega\uparrow\omega$) [$\omega\uparrow\omega$,α] *2*
⟨*proof*⟩

end

4 Acknowledgements

The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council. Many thanks to Mirna Džamonja (who suggested the project) and Angeliki Koutsoukou-Argyraki for assistance at tricky moments.

References

[1] P. Erdös and E. C. Milner. A theorem in the partition calculus. *Canadian Mathematical Bulletin*, 15(4):501–505, Dec. 1972.

[2] P. Erdös and E. C. Milner. A theorem in the partition calculus corrigendum. *Canadian Mathematical Bulletin*, 17(2):305, June 1974.

[3] J. A. Larson. A short proof of a partition theorem for the ordinal ω^ω. *Annals of Mathematical Logic*, 6(2):129–145, Dec. 1973.