

# A Partition Theorem for the Ordinal $\omega^\omega$

Lawrence C. Paulson

February 6, 2026

## Abstract

The theory of partition relations concerns generalisations of Ramsey's theorem. For any ordinal  $\alpha$ , write  $\alpha \rightarrow (\alpha, m)^2$  if for each function  $f$  from unordered pairs of elements of  $\alpha$  into  $\{0, 1\}$ , either there is a subset  $X \subseteq \alpha$  order-isomorphic to  $\alpha$  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  $\omega \rightarrow (\omega, \omega)^2$ , or if we restrict  $m$  to the positive integers as above, then  $\omega \rightarrow (\omega, m)^2$  for all  $m$  [3].

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

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

<b>1</b>	<b>Library additions</b>	<b>2</b>
1.1	Other material . . . . .	3
1.2	The list-of function . . . . .	4
1.3	Monotonic enumeration of a countably infinite set . . . . .	4
<b>2</b>	<b>Ordinal Partitions</b>	<b>6</b>
2.1	Ordinal Partitions: Definitions . . . . .	6
2.2	Relating partition properties on $VWF$ to the general case . . . . .	8
2.3	Simple consequences of the definitions . . . . .	8
2.4	Specker's theorem . . . . .	8
2.5	Erds-Milner theorem . . . . .	10
<b>3</b>	<b>An ordinal partition theorem by Jean A. Larson</b>	<b>11</b>
3.1	Cantor normal form for ordinals below $\omega \uparrow \omega$ . . . . .	11
3.2	Larson's set $W(n)$ . . . . .	13
3.3	Definitions required for the lemmas . . . . .	15

3.3.1	Larson's " $<$ "-relation on ordered lists . . . . .	15
3.4	Nash Williams for lists . . . . .	16
3.4.1	Thin sets of lists . . . . .	16
3.5	Specialised functions on lists . . . . .	17
3.6	Forms and interactions . . . . .	19
3.6.1	Forms . . . . .	19
3.6.2	Interactions . . . . .	20
3.6.3	Injectivity of interactions . . . . .	21
3.7	For Lemma 3.8 AND PROBABLY 3.7 . . . . .	22
3.8	Larson's Lemma 3.11 . . . . .	23
3.9	Larson's Lemma 3.6 . . . . .	23
3.10	Larson's Lemma 3.7 . . . . .	23
3.10.1	Preliminaries . . . . .	23
3.10.2	Lemma 3.7 of Jean A. Larson, <i>ibid.</i> . . . . .	24
3.11	Larson's Lemma 3.8 . . . . .	24
3.11.1	Primitives needed for the inductive construction of $b$ . . . . .	24
3.11.2	Special primitives for the ordertype proof . . . . .	25
3.11.3	The final part of 3.8, where two sequences are merged . . . . .	25
3.11.4	Actual proof of Larson's Lemma 3.8 . . . . .	27
3.12	The main partition theorem for $\omega \uparrow \omega$ . . . . .	28

**4 Acknowledgements** **28**

# 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  $i: i < card\ S$ 
  shows  $enumerate\ (S - \{x\})\ i = enumerate\ S\ i$ 
  <proof>
```

```
lemma hd-lex:  $\llbracket hd\ ms < hd\ ns; length\ ms = length\ ns; ns \neq [] \rrbracket \implies (ms, ns) \in lex\ less-than$ 
  <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*  $x \in \text{list.set } xs$   
**shows**  $x \leq \text{last } xs$   
 $\langle \text{proof} \rangle$

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

**lemma** *sorted-hd-le-last*:  
**assumes** *sorted xs*  $xs \neq []$   
**shows**  $\text{hd } xs \leq \text{last } xs$   
 $\langle \text{proof} \rangle$

**lemma** *sorted-list-of-set-set-of [simp]*:  $\text{strict-sorted } l \implies \text{sorted-list-of-set } (\text{list.set } l) = l$   
 $\langle \text{proof} \rangle$

**lemma** *range-strict-mono-ext*:  
**fixes**  $f :: \text{nat} \Rightarrow 'a :: \text{linorder}$   
**assumes**  $\text{eq} : \text{range } f = \text{range } g$   
**and**  $\text{sm} : \text{strict-mono } f \text{ strict-mono } g$   
**shows**  $f = g$   
 $\langle \text{proof} \rangle$

## 1.1 Other material

**definition** *strict-mono-sets* ::  $['a :: \text{order set}, 'b :: \text{order set}] \Rightarrow \text{bool}$  **where**  
 $\text{strict-mono-sets } A f \equiv \forall x \in A. \forall y \in A. x < y \longrightarrow \text{less-sets } (f x) (f y)$

**lemma** *strict-mono-setsD*:  
**assumes**  $\text{strict-mono-sets } A f$   $x < y$   $x \in A$   $y \in A$   
**shows**  $\text{less-sets } (f x) (f y)$   
 $\langle \text{proof} \rangle$

**lemma** *strict-mono-sets-imp-disjoint*:  
**fixes**  $A :: 'a :: \text{linorder set}$   
**assumes**  $\text{strict-mono-sets } A f$   
**shows**  $\text{pairwise } (\lambda x y. \text{disjnt } (f x) (f y)) A$   
 $\langle \text{proof} \rangle$

**lemma** *strict-mono-sets-subset*:  
**assumes**  $\text{strict-mono-sets } B f$   $A \subseteq B$   
**shows**  $\text{strict-mono-sets } A f$   
 $\langle \text{proof} \rangle$

**lemma** *strict-mono-less-sets-Min*:  
**assumes**  $\text{strict-mono-sets } I f$  *finite I*  $I \neq \{\}$   
**shows**  $\text{less-sets } (f (\text{Min } I)) (\bigcup (f ` (I - \{\text{Min } I\})))$

*<proof>*

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

**lemma** *infinite-finite-Inter*:  
**assumes** *finite*  $\mathcal{A}$   $\mathcal{A} \neq \{\}$   $\bigwedge A. A \in \mathcal{A} \implies \text{infinite } A$   
**and**  $\bigwedge A B. \llbracket A \in \mathcal{A}; B \in \mathcal{A} \rrbracket \implies A \cap B \in \mathcal{A}$   
**shows** *infinite*  $(\bigcap \mathcal{A})$   
*<proof>*

**lemma** *atLeast-less-sets*:  $\llbracket \text{less-sets } A \{x\}; B \subseteq \{x..\} \rrbracket \implies \text{less-sets } A B$   
*<proof>*

## 1.2 The list-of function

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

**lemma** *sorted-list-of-set-Un*:  
**assumes**  $AB$ : *less-sets*  $A B$  **and** *fin*: *finite*  $A$  *finite*  $B$   
**shows** *sorted-list-of-set*  $(A \cup B)$  = *sorted-list-of-set*  $A$  @ *sorted-list-of-set*  $B$   
*<proof>*

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

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

## 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**  $N :: \text{nat set}$   
**shows** *infinite*  $N \longleftrightarrow (\exists f::\text{nat} \Rightarrow \text{nat}. N = \text{range } f \wedge \text{strict-mono } f)$   
*<proof>*

**lemma** *enum-works*:  
**fixes**  $N :: \text{nat set}$   
**assumes** *infinite*  $N$   
**shows**  $N = \text{range } (\text{enum } N) \wedge \text{strict-mono } (\text{enum } N)$   
 $\langle \text{proof} \rangle$

**lemma** *range-enum*:  $\text{range } (\text{enum } N) = N$  **and** *strict-mono-enum*: *strict-mono*  $(\text{enum } N)$   
**if** *infinite*  $N$  **for**  $N :: \text{nat set}$   
 $\langle \text{proof} \rangle$

**lemma** *enum-0-eq-Inf*:  
**fixes**  $N :: \text{nat set}$   
**assumes** *infinite*  $N$   
**shows**  $\text{enum } N 0 = \text{Inf } N$   
 $\langle \text{proof} \rangle$

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

**lemma** *enum-obtain-index-finite*:  
**fixes**  $N :: \text{nat set}$   
**assumes**  $x \in N$  *finite*  $N$   
**obtains**  $i$  **where**  $i < \text{card } N$   $x = \text{enum } N i$   
 $\langle \text{proof} \rangle$

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

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

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

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

*<proof>*

**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*}  $\longleftrightarrow$  ( $\forall x \in A. x < a$ )  
*<proof>*

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

**lemma** *less-sets-imp-strict-mono-sets*:  
**assumes**  $\bigwedge i. \text{less-sets } (A \ i) \ (A \ (\text{Suc } i)) \ \wedge i. i > 0 \implies A \ i \neq \{\}$   
**shows** *strict-mono-sets UNIV A*  
*<proof>*

**lemma** *less-sets-Suc-Max*:  
**assumes** *finite A*  
**shows** *less-sets A {Suc (Max A)..}*  
*<proof>*

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

**end**

## 2 Ordinal Partitions

Material from Jean A. Larson, A short proof of a partition theorem for the ordinal  $\omega^\omega$ . *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-1st* :: [*'a*  $\times$  *'a*] *set*, *'a set*, *V list*, *nat*]  $\Rightarrow$  *bool*  
**where** *partn-1st r B  $\alpha$  n*  $\equiv$   $\forall f \in [B]^n \rightarrow \{.. $\text{length } \alpha$ \}. \exists i < \text{length } \alpha. \exists H. H \subseteq B \wedge \text{ordertype } H \ r = (\alpha!i) \wedge f' (nsets \ H \ n) \subseteq \{i\}$

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

**lemma** *partn-lst-E*:

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

**lemma** *partn-lst-VWF-nontriv*:

**assumes** *partn-lst-VWF*  $\beta \alpha n l = \text{length } \alpha \text{ Ord } \beta l > 0$   
**obtains**  $i$  **where**  $i < l \alpha!i \leq \beta$   
*<proof>*

**lemma** *partn-lst-triv0*:

**assumes**  $\alpha!i = 0 i < \text{length } \alpha n \neq 0$   
**shows** *partn-lst*  $r B \alpha n$   
*<proof>*

**lemma** *partn-lst-triv1*:

**assumes**  $\alpha!i \leq 1 i < \text{length } \alpha n > 1 B \neq \{\}$  *wf*  $r$   
**shows** *partn-lst*  $r B \alpha n$   
*<proof>*

**lemma** *partn-lst-two-swap*:

**assumes** *partn-lst*  $r B [x,y] n$  **shows** *partn-lst*  $r B [y,x] n$   
*<proof>*

**lemma** *partn-lst-greater-resource*:

**assumes**  $M$ : *partn-lst*  $r B \alpha n$  **and**  $B \subseteq C$   
**shows** *partn-lst*  $r C \alpha n$   
*<proof>*

**lemma** *partn-lst-less*:

**assumes**  $M$ : *partn-lst*  $r B \alpha n$  **and**  $\text{eq: length } \alpha' = \text{length } \alpha$  **and**  $\text{List.set } \alpha' \subseteq \text{ON}$   
**and**  $le$ :  $\bigwedge i. i < \text{length } \alpha \implies \alpha!i \leq \alpha!i$   
**and**  $r$ : *wf*  $r \text{ trans } r \text{ total-on } B r$  **and** *small*  $B$   
**shows** *partn-lst*  $r B \alpha' n$   
*<proof>*

Holds because no  $n$ -sets exist!

**lemma** *partn-lst-VWF-degenerate*:

**assumes**  $k < n$   
**shows** *partn-lst-VWF*  $\omega (\text{ord-of-nat } k \# \alpha s) n$   
*<proof>*

**lemma** *partn-lst-VWF- $\omega$ -2*:

**assumes** *Ord*  $\alpha$

**shows** *partn-lst-VWF*  $(\omega \uparrow (1+\alpha)) [2, \omega \uparrow (1+\alpha)] 2$  (**is** *partn-lst-VWF*  $? \beta$  - -)  
*<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*  $r U \alpha n$  **and**  $\beta$ : *ordertype*  $U r = \beta$  **and** *small*  $U$

**and**  $r$ : *wf*  $r$  *trans*  $r$  *total-on*  $U r$

**shows** *partn-lst-VWF*  $\beta \alpha n$

*<proof>*

**lemma** *partn-lst-imp-partn-lst-VWF*:

**assumes** *part*: *partn-lst*  $r U \alpha n$  **and**  $\beta$ : *ordertype*  $U r \leq \beta$  *small*  $U$

**and**  $r$ : *wf*  $r$  *trans*  $r$  *total-on*  $U r$

**shows** *partn-lst-VWF*  $\beta \alpha n$

*<proof>*

**lemma** *partn-lst-VWF-imp-partn-lst-eq*:

**assumes** *part*: *partn-lst-VWF*  $\beta \alpha n$  **and**  $\beta$ : *ordertype*  $U r = \beta$  *small*  $U$

**and**  $r$ : *wf*  $r$  *trans*  $r$  *total-on*  $U r$

**shows** *partn-lst*  $r U \alpha n$

*<proof>*

**corollary** *partn-lst-VWF-imp-partn-lst*:

**assumes** *partn-lst-VWF*  $\beta \alpha n$  **and**  $\beta$ : *ordertype*  $U r \geq \beta$  *small*  $U$

*wf*  $r$  *trans*  $r$  *total-on*  $U r$

**shows** *partn-lst*  $r U \alpha n$

*<proof>*

## 2.3 Simple consequences of the definitions

A restatement of the infinite Ramsey theorem using partition notation

**lemma** *Ramsey-partn*: *partn-lst-VWF*  $\omega [\omega, \omega] 2$

*<proof>*

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

**proposition** *omega-basic-counterexample*:

**assumes** *Ord*  $\alpha$

**shows**  $\neg$  *partn-lst-VWF*  $\alpha [succ (vcard \alpha), \omega] 2$

*<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*  $\equiv \exists a b c d. u = \{(a,b),(c,d)\} \wedge \text{form-split } a b c d i$

**definition** *scheme* :: [(nat\*nat)set]  $\Rightarrow$  nat set **where**  
*scheme* *u*  $\equiv \text{fst } ' u \cup \text{snd } ' u$

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

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

**lemma** *ordertype-UU-ge- $\omega$ 2*: ordertype UNIV pair-less  $\leq$  ordertype UU pair-less  
 ⟨proof⟩

**lemma** *ordertype-UU- $\omega$ 2*: 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  $\omega^\omega$ . *Annals of Mathematical Logic*, 6:129–145, 1973.

**lemma** *lemma-2-3*:

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

Lemma 2.4 of Jean A. Larson, *ibid.*

**lemma** *lemma-2-4*:

**assumes** infinite *N* *k* < 4  
**obtains** *M* **where**  $M \in [UU]^m \wedge u. u \in [M]^2 \implies \text{form } u k \wedge u. u \in [M]^2 \implies \text{scheme } u \subseteq N$   
 ⟨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 =  $\omega \uparrow 2$   
 $\bigwedge u. u \in [X]^2 \implies (\exists k < 4. \text{form } u k) \wedge \text{scheme } u \subseteq N$   
 ⟨proof⟩

Theorem 2.1 of Jean A. Larson, *ibid.*

**lemma** *Specker-aur*:

**assumes**  $\alpha \in \text{elts } \omega$   
**shows** *partn-lst* pair-less UU [ $\omega \uparrow 2, \alpha$ ] 2  
 ⟨proof⟩

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

**end**

**theory** *Erdos-Milner*

**imports** *Partitions*

**begin**

## 2.5 Erdos-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**  $D: D \subseteq \text{elts } \beta$  **and**  $\text{Ord } \beta$

**obtains**  $L$  **where**  $\bigcup (\text{List.set } L) = D \wedge X. X \in \text{List.set } L \implies \text{indecomposable } (tp\ X)$

**and**  $\bigwedge M. \llbracket M \subseteq D; \bigwedge X. X \in \text{List.set } L \implies tp\ (M \cap X) \geq tp\ X \rrbracket \implies tp\ M = tp\ D$

⟨*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*  $A \subseteq ON$

**and**  $x \in A$  **and**  $A1: tp\ A1 = \alpha$   $A1 \subseteq A$

**obtains**  $A2$  **where**  $tp\ A2 = \alpha$   $A2 \subseteq A1 \setminus \{x\} \ll A2$

⟨*proof*⟩

the main theorem, from which they derive the headline result

**theorem** *Erdos-Milner-aux*:

**assumes** *part*: *partn-lst-VWF*  $\alpha [k, \gamma] 2$

**and** *indec*: *indecomposable*  $\alpha$  **and**  $k > 1$  *Ord*  $\gamma$  **and**  $\beta: \beta \in \text{elts } \omega 1$

**shows** *partn-lst-VWF*  $(\alpha * \beta) [\text{ord-of-nat } (2 * k), \text{min } \gamma (\omega * \beta)] 2$

⟨*proof*⟩

**theorem** *Erdos-Milner*:

**assumes**  $\nu: \nu \in \text{elts } \omega 1$

**shows** *partn-lst-VWF*  $(\omega \uparrow (1 + \nu * n)) [\text{ord-of-nat } (2 \hat{\sim} n), \omega \uparrow (1 + \nu)] 2$

⟨*proof*⟩

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

Theorem 3.2 of Jean A. Larson, *ibid.*

**corollary** *Theorem-3-2:*

**fixes**  $k n :: \text{nat}$

**shows** *partn-lst-VWF*  $(\omega^\uparrow(n*k)) [\omega^\uparrow n, \text{ord-of-nat } k] 2$   
 ⟨*proof*⟩

**end**

### 3 An ordinal partition theorem by Jean A. Larson

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

**theory** *Omega-Omega*

**imports** *HOL-Library.Product-Lexorder Erdos-Milner*

**begin**

**abbreviation** *list-of*  $\equiv$  *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*  $(\text{Suc } n) [] = 0$

| *Cons: omega-sum-aux*  $(\text{Suc } n) (m\#ms) = (\omega^\uparrow n) * (\text{ord-of-nat } m) + \text{omega-sum-aux } n \ ms$

**abbreviation** *omega-sum* **where** *omega-sum*  $ms \equiv \text{omega-sum-aux } (\text{length } ms) \ ms$

A normal expansion has no leading zeroes

**inductive** *normal:: nat list*  $\Rightarrow$  *bool* **where**

*normal-Nil*[*iff*]: *normal*  $[]$

| *normal-Cons*:  $m > 0 \implies \text{normal } (m\#ms)$

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

**lemma** *omega-sum-0-iff* [*simp*]: *normal*  $ns \implies \text{omega-sum } ns = 0 \iff ns = []$

⟨*proof*⟩

**lemma** *Ord-omega-sum-aux* [*simp*]: *Ord*  $(\text{omega-sum-aux } k \ ms)$

*<proof>*

**lemma** *Ord-omega-sum: Ord (omega-sum ms)*

*<proof>*

**lemma** *omega-sum-less- $\omega\omega$  [intro]: omega-sum ms <  $\omega \uparrow \omega$*

*<proof>*

**lemma** *omega-sum-aux-less: omega-sum-aux k ms <  $\omega \uparrow k$*

*<proof>*

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

*<proof>*

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

*<proof>*

**lemma** *omega-sum-length-less:*

**assumes** *normal ns length ms < length ns*

**shows** *omega-sum ms < omega-sum ns*

*<proof>*

**lemma** *omega-sum-length-leD:*

**assumes** *omega-sum ms  $\leq$  omega-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-sum (m#ms) < omega-sum (n#ns)  $\iff m < n \vee m = n \wedge \text{omega-sum ms} < \text{omega-sum ns}$*

*<proof>*

**lemma** *omega-sum-less-iff-cases:*

**assumes**  *$m > 0 \ n > 0$*

**shows** *omega-sum (m#ms) < omega-sum (n#ns)*

$\iff$  *length ms < length ns*

$\vee$  *length ms = length ns  $\wedge$   $m < n$*

$\vee$  *length ms = length ns  $\wedge$   $m = n \wedge \text{omega-sum ms} < \text{omega-sum ns}$*

*<proof>*

**lemma** *omega-sum-less-iff:*

*((length ms, omega-sum ms), (length ns, omega-sum ns))  $\in$  less-than  $\langle *lex* \rangle$*   
*VWF*

$\iff$  *(ms, ns)  $\in$  lenlex less-than*

*<proof>*

**lemma** *eq-omega-sum-less-iff:*

**assumes** *length ms = length ns*

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

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

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

**lemma** *Ex-omega-sum*:  $\gamma \in \text{elts } (\omega \uparrow n) \implies \exists ns. \gamma = \text{omega-sum } ns \wedge \text{length } ns = n$   
 <proof>

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

**lemma** *normal-drop [simp]*:  $\text{normal } (\text{dropWhile } (\lambda n. n=0) ns)$   
 <proof>

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

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

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

### 3.2 Larson's set $W(n)$

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

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

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

**lemma** *from-into-WW* [simp]:  $\text{from-WW } k \ (\text{into-WW } k \ ns) = ns$   
(proof)

**lemma** *inj-into-WW*:  $\text{inj} \ (\text{into-WW } k)$   
(proof)

**lemma** *into-from-WW-aux*:  
[[*strict-sorted*  $ns$ ;  $\forall n \in \text{list.set } ns. k \leq n$ ]]  $\implies \text{into-WW } k \ (\text{from-WW } k \ ns) = ns$   
(proof)

**lemma** *into-from-WW* [simp]:  $\text{strict-sorted } ns \implies \text{into-WW } 0 \ (\text{from-WW } 0 \ ns) = ns$   
(proof)

**lemma** *into-WW-imp-ge*:  $y \in \text{List.set} \ (\text{into-WW } x \ ns) \implies x \leq y$   
(proof)

**lemma** *strict-sorted-into-WW*:  $\text{strict-sorted} \ (\text{into-WW } x \ ns)$   
(proof)

**lemma** *length-into-WW*:  $\text{length} \ (\text{into-WW } x \ ns) = \text{length } ns$   
(proof)

**lemma** *WW-eq-range-into*:  $WW = \text{range} \ (\text{into-WW } 0)$   
(proof)

**lemma** *into-WW-lenlex-iff*:  $(\text{into-WW } k \ ms, \text{into-WW } k \ ns) \in \text{lenlex less-than} \iff (ms, ns) \in \text{lenlex less-than}$   
(proof)

**lemma** *wf-llt* [simp]:  $\text{wf} \ (\text{lenlex less-than})$  **and** *trans-llt* [simp]:  $\text{trans} \ (\text{lenlex less-than})$   
(proof)

**lemma** *total-llt* [simp]:  $\text{total-on } A \ (\text{lenlex less-than})$   
(proof)

**lemma** *omega-sum-1-less*:  
**assumes**  $(ms, ns) \in \text{lenlex less-than}$  **shows**  $\text{omega-sum} \ (1 \# ms) < \text{omega-sum} \ (1 \# ns)$   
(proof)

**lemma** *ordertype-WW-1*:  $\text{ordertype } WW \ (\text{lenlex less-than}) \leq \text{ordertype } UNIV \ (\text{lenlex less-than})$   
(proof)

**lemma** *ordertype-WW-2*:  $\text{ordertype } UNIV \ (\text{lenlex less-than}) \leq \omega \uparrow \omega$   
(proof)

**lemma** *ordertype-WW-3*:  $\omega \uparrow \omega \leq \text{ordertype } WW \ (\text{lenlex less-than})$

*<proof>*

**lemma** *ordertype-WW*: *ordertype WW (lenlex less-than) =  $\omega \uparrow \omega$*   
**and** *ordertype-UNIV- $\omega\omega$* : *ordertype UNIV (lenlex less-than) =  $\omega \uparrow \omega$*   
*<proof>*

**lemma** *ordertype- $\omega\omega$* :  
**fixes** *F :: nat  $\Rightarrow$  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  $\Rightarrow$  nat list set*  
**where** *WW-seg n  $\equiv$  {l  $\in$  WW. length l = n}*

**lemma** *WW-seg-subset-WW*: *WW-seg n  $\subseteq$  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  $\leq$  ys  $\equiv$  xs < ys  $\vee$  xs = ys* **for** *xs ys :: 'a list*

**instance**  
*<proof>*

**end**

**lemma** *less-Nil [simp]*: *xs < []  $\wedge$  [] < xs*  
*<proof>*

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

**assumes**  $list.set\ xs \ll list.set\ ys$   
**shows**  $xs < ys$   
 $\langle proof \rangle$

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

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

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

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

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

**lemma** *less-hd-imp-less*:  $xs < [hd\ ys] \implies xs < ys$   
 $\langle proof \rangle$

**lemma** *strict-sorted-concat-I*:  
**assumes**  $\bigwedge x. x \in list.set\ xs \implies strict-sorted\ x$   
 $\bigwedge n. Suc\ n < length\ xs \implies xs!n < xs!Suc\ n$   
 $xs \in lists\ (-\ \{\}\}$   
**shows**  $strict-sorted\ (concat\ xs)$   
 $\langle proof \rangle$

## 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**  $ys \neq [] \wedge hd\ ys = hd\ xs$   
 $\langle proof \rangle$

**lemma** *take-initial-segment*:

**assumes** *initial-segment xs ys k ≤ 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) ↔ x=y ∧ 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) ↔ initial-segment xs ys (is ?lhs = ?rhs)*

*<proof>*

**theorem** *Nash-Williams-WW*:

**fixes** *h :: nat list ⇒ nat*

**assumes** *infinite M and h: h ‘ {l ∈ A. List.set l ⊆ M} ⊆ {..<2} and thin A A ⊆ WW*

**obtains** *i N where i < 2 infinite N N ⊆ M h ‘ {l ∈ A. List.set l ⊆ N} ⊆ {i}*

*<proof>*

### 3.5 Specialised functions on lists

**lemma** *mem-lists-non-Nil: xss ∈ lists (– {[]}) ↔ (∀ x ∈ list.set xss. x ≠ [])*

*<proof>*

**fun** *acc-lengths :: nat ⇒ 'a list list ⇒ 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 = [] ↔ ls = []*

*<proof>*

**lemma** *set-acc-lengths*:

**assumes** *ls ∈ lists (– {[]}) shows list.set (acc-lengths acc ls) ⊆ {acc<..}*

*<proof>*

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

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

**lemma** *last-acc-lengths* [*simp*]:  
 $ls \neq [] \implies last (acc-lengths\ acc\ ls) = acc + sum-list (map\ length\ ls)$   
 ⟨*proof*⟩

**lemma** *nth-acc-lengths* [*simp*]:  
 $\llbracket ls \neq []; k < length\ ls \rrbracket \implies acc-lengths\ acc\ ls\ !\ k = acc + sum-list (map\ length\ (take\ (Suc\ k)\ ls))$   
 ⟨*proof*⟩

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

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

**lemma** *length-concat-acc-lengths*:  
 $ls \neq [] \implies k + length (concat\ ls) \in list.set (acc-lengths\ k\ ls)$   
 ⟨*proof*⟩

**lemma** *strict-sorted-acc-lengths*:  
**assumes**  $ls \in lists\ (-\ \{\!\!\}\! \})$  **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**  $as \in lists\ (-\ \{\!\!\}\! \})$   
**shows**  $length (concat\ as) \geq length\ as$   
 ⟨*proof*⟩

**fun** *interact* :: 'a list list  $\Rightarrow$  'a list list  $\Rightarrow$  '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*:  
 $length (interact\ xs\ ys) = sum-list (map\ length\ xs) + sum-list (map\ length\ ys)$

*<proof>*

**lemma** *length-interact-ge*:

**assumes**  $xs \in \text{lists } (- \{\{\}\})$   $ys \in \text{lists } (- \{\{\}\})$

**shows**  $\text{length } (\text{interact } xs \ ys) \geq \text{length } xs + \text{length } ys$

*<proof>*

**lemma** *set-interact [simp]*:

**shows**  $\text{list.set } (\text{interact } xs \ ys) = \text{list.set } (\text{concat } xs) \cup \text{list.set } (\text{concat } ys)$

*<proof>*

**lemma** *interact-eq-Nil-iff [simp]*:

**assumes**  $xs \in \text{lists } (- \{\{\}\})$   $ys \in \text{lists } (- \{\{\}\})$

**shows**  $\text{interact } xs \ ys = [] \iff xs=[] \wedge ys=[]$

*<proof>*

**lemma** *interact-sing [simp]*:  $\text{interact } [x] \ ys = x \ @ \ \text{concat } ys$

*<proof>*

**lemma** *hd-interact*:  $[[xs \neq []; \text{hd } xs \neq []]] \implies \text{hd } (\text{interact } xs \ ys) = \text{hd } (\text{hd } xs)$

*<proof>*

**lemma** *acc-lengths-concat-injective*:

**assumes**  $\text{concat } as' = \text{concat } as$   $\text{acc-lengths } n \ as' = \text{acc-lengths } n \ as$

**shows**  $as' = as$

*<proof>*

**lemma** *acc-lengths-interact-injective*:

**assumes**  $\text{interact } as' \ bs' = \text{interact } as \ bs$   $\text{acc-lengths } a \ as' = \text{acc-lengths } a \ as$   
 $\text{acc-lengths } b \ bs' = \text{acc-lengths } b \ bs$

**shows**  $as' = as \wedge bs' = bs$

*<proof>*

**lemma** *strict-sorted-interact-I*:

**assumes**  $\text{length } ys \leq \text{length } xs$   $\text{length } xs \leq \text{Suc } (\text{length } ys)$

$\bigwedge x. x \in \text{list.set } xs \implies \text{strict-sorted } x$

$\bigwedge y. y \in \text{list.set } ys \implies \text{strict-sorted } y$

$\bigwedge n. n < \text{length } ys \implies xs!n < ys!n$

$\bigwedge n. \text{Suc } n < \text{length } xs \implies ys!n < xs!\text{Suc } n$

**assumes**  $xs \in \text{lists } (- \{\{\}\})$   $ys \in \text{lists } (- \{\{\}\})$

**shows**  $\text{strict-sorted } (\text{interact } xs \ ys)$

*<proof>*

## 3.6 Forms and interactions

### 3.6.1 Forms

**inductive** *Form-Body* ::  $[\text{nat}, \text{nat}, \text{nat list}, \text{nat list}, \text{nat list}] \Rightarrow \text{bool}$

**where** *Form-Body*  $ka \ kb \ xs \ ys \ zs$

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

**inductive**  $\text{Form} :: [\text{nat}, \text{nat list set}] \Rightarrow \text{bool}$   
**where**  $\text{Form } 0 \{xs,ys\}$  **if**  $\text{length } xs = \text{length } ys$   $xs \neq ys$   
|  $\text{Form } (2*k-1) \{xs,ys\}$  **if**  $\text{Form-Body } k k xs ys zs k > 0$   
|  $\text{Form } (2*k) \{xs,ys\}$  **if**  $\text{Form-Body } (\text{Suc } k) k xs ys zs k > 0$

**inductive-cases**  $\text{Form-0-cases-raw}: \text{Form } 0 u$

**lemma** *Form-elim-upair*:  
**assumes**  $\text{Form } l U$   
**obtains**  $xs ys$  **where**  $xs \neq ys$   $U = \{xs,ys\}$   $\text{length } xs \leq \text{length } ys$   
*<proof>*

**lemma** **assumes**  $\text{Form-Body } ka kb xs ys zs$   
**shows**  $\text{Form-Body-WW}: zs \in WW$   
**and**  $\text{Form-Body-nonempty}: \text{length } zs > 0$   
**and**  $\text{Form-Body-length}: \text{length } xs < \text{length } ys$   
*<proof>*

**lemma** *form-cases*:  
**fixes**  $l::\text{nat}$   
**obtains**  $(\text{zero}) l = 0$  |  $(\text{nz}) ka kb$  **where**  $l = ka+kb - 1$   $0 < kb$   $kb \leq ka$   $ka \leq \text{Suc } kb$   
*<proof>*

### 3.6.2 Interactions

**lemma** *interact*:  
**assumes**  $\text{Form } l U l > 0$   
**obtains**  $ka kb xs ys zs$  **where**  $l = ka+kb - 1$   $U = \{xs,ys\}$   $\text{Form-Body } ka kb xs$   
 $ys zs$   $0 < kb$   $kb \leq ka$   $ka \leq \text{Suc } kb$   
*<proof>*

**definition** *inter-scheme* ::  $\text{nat} \Rightarrow \text{nat list set} \Rightarrow \text{nat list}$   
**where**  $\text{inter-scheme } l U \equiv$   
 $\text{SOME } zs. \exists k xs ys. U = \{xs,ys\} \wedge$   
 $(l = 2*k-1 \wedge \text{Form-Body } k k xs ys zs \vee l = 2*k \wedge \text{Form-Body } (\text{Suc } k)$   
 $k xs ys zs)$

**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*:

$strict-sorted\ (interact\ as\ bs) \implies strict-sorted\ (concat\ as) \wedge strict-sorted\ (concat\ bs)$

(*proof*)

**lemma** *strict-sorted-interact-hd*:

$[strict-sorted\ (interact\ cs\ ds); cs \neq []; ds \neq []; hd\ cs \neq []; hd\ ds \neq []]$   
 $\implies hd\ (hd\ cs) < hd\ (hd\ ds)$

(*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**  $zs' = zs$   
 $\langle proof \rangle$

**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**  $zs = inter\ scheme\ ((ka+kb) - Suc\ 0)\ \{xs,ys\}$   
 $\langle proof \rangle$

### 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]:*  $grab\ N\ 0 = (\{\}, N)$   
 $\langle proof \rangle$

**lemma** *less-sets-grab:*

$infinite\ N \Longrightarrow fst\ (grab\ N\ n) \ll snd\ (grab\ N\ n)$   
 $\langle proof \rangle$

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

**lemma** *card-grab [simp]:*

**assumes**  $infinite\ N$  **shows**  $card\ (fst\ (grab\ N\ n)) = n$   
 $\langle proof \rangle$

**lemma** *fst-grab-subset:*  $fst\ (grab\ N\ n) \subseteq N$   
 $\langle proof \rangle$

**lemma** *snd-grab-subset:*  $snd\ (grab\ N\ n) \subseteq N$   
 $\langle proof \rangle$

**lemma** *grab-Un-eq:*

**assumes**  $infinite\ N$  **shows**  $fst\ (grab\ N\ n) \cup snd\ (grab\ N\ n) = N$   
 $\langle proof \rangle$

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

**lemma** *grab-eqD:*

$\llbracket grab\ N\ n = (A,M); infinite\ N \rrbracket$   
 $\Longrightarrow A \ll M \wedge finite\ A \wedge card\ A = n \wedge infinite\ M \wedge A \subseteq N \wedge M \subseteq N$   
 $\langle proof \rangle$

**lemma** *less-sets-fst-grab*:  $A \ll N \implies A \ll \text{fst } (\text{grab } N \ n)$   
 ⟨proof⟩

Possibly redundant, given *grab*

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

**lemma** *infinite-nextN*:  $\text{infinite } N \implies \text{infinite } (\text{next } N \ n)$   
 ⟨proof⟩

**lemma** *next-subset*:  $\text{next } N \ n \subseteq N$   
 ⟨proof⟩

**lemma** *next-subset-greaterThan*:  $m \leq n \implies \text{next } N \ n \subseteq \{m<..\}$   
 ⟨proof⟩

**lemma** *next-subset-atLeast*:  $m \leq n \implies \text{next } N \ n \subseteq \{m..\}$   
 ⟨proof⟩

**lemma** *enum-next-ge*:  $\text{infinite } N \implies a \leq \text{enum } (\text{next } N \ a) \ n$   
 ⟨proof⟩

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

### 3.8 Larson's Lemma 3.11

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

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

### 3.9 Larson's Lemma 3.6

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

*<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**  $M \in [WW]^m$

$\bigwedge U.\ U \in [M]^2 \implies Form\ l\ U \wedge List.set\ (inter-scheme\ l\ U) \subseteq N$

*<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*:  $u \in IJ\ k \longleftrightarrow (\exists j\ i.\ u = (j,i) \wedge i < j \wedge j \leq k)$

*<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*:  $prev\ j\ i = None \longleftrightarrow j \leq Suc\ 0 \wedge i = 0$

*<proof>*

**lemma** *prev-pair-less*:

$prev\ j\ i = Some\ ji' \implies (ji', (j,i)) \in pair-less$   
 ⟨proof⟩

**lemma** *prev-Some-less*:  $\llbracket prev\ j\ i = Some\ (j',i');\ i \leq j \rrbracket \implies i' < j'$   
 ⟨proof⟩

**lemma** *prev-maximal*:  
 $\llbracket prev\ j\ i = Some\ (j',i');\ (ji'', (j,i)) \in pair-less;\ ji'' \in IJ\ k \rrbracket$   
 $\implies (ji'', (j',i')) \in pair-less \vee ji'' = (j',i')$   
 ⟨proof⟩

**lemma** *pair-less-prev*:  
 assumes  $(u, (j,i)) \in pair-less\ u \in IJ\ k$   
 shows  $prev\ j\ i = Some\ u \vee (\exists x. (u, x) \in pair-less \wedge prev\ j\ i = Some\ x)$   
 ⟨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  $USigma\ A\ B \equiv \bigcup X \in A. \bigcup y \in B\ X. \{insert\ y\ X\}$

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

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

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

**proposition** *ordertype-append-image-IJ*:  
 assumes  $lenB\ [simp]: \bigwedge i\ j. i \in \mathcal{I} \implies j \in J\ i \implies length\ (B\ j) = c$   
 and  $AB: \bigwedge i\ j. i \in \mathcal{I} \implies j \in J\ i \implies A\ i < B\ j$   
 and  $IJ: \bigwedge i. i \in \mathcal{I} \implies i \ll J\ i \wedge finite\ i$   
 and  $\beta: \bigwedge i. i \in \mathcal{I} \implies ordertype\ (B\ 'J\ i)\ (lenlex\ less-than) = \beta$   
 and  $A: inj-on\ A\ \mathcal{I}$   
 shows  $ordertype\ (usplit\ (\lambda i\ j. A\ i @ B\ j)\ 'USigma\ \mathcal{I}\ J)\ (lenlex\ less-than)$   
 $= \beta * ordertype\ (A\ ' \mathcal{I})\ (lenlex\ less-than)$   
 (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  $NullNull: merge\ []\ []\ []\ []$   
 |  $Null: as \neq [] \implies merge\ as\ []\ [concat\ as]\ []$   
 |  $App: \llbracket as1 \neq [];\ bs1 \neq [] \rrbracket;$

$\text{concat } as1 < \text{concat } bs1; \text{concat } bs1 < \text{concat } as2; \text{merge } as2 \text{ } bs2 \text{ } as$   
 $bs]$   
 $\implies \text{merge } (as1@as2) (bs1@bs2) (\text{concat } as1 \# as) (\text{concat } bs1 \# bs)$

**inductive-simps** *Null1* [*simp*]:  $\text{merge } [] \text{ } bs \text{ } us \text{ } vs$

**inductive-simps** *Null2* [*simp*]:  $\text{merge } as \text{ } [] \text{ } us \text{ } vs$

**lemma** *merge-single*:

$\llbracket \text{concat } as < \text{concat } bs; \text{concat } as \neq []; \text{concat } bs \neq [] \rrbracket \implies \text{merge } as \text{ } bs \text{ } [\text{concat } as]$   
 $[\text{concat } bs]$   
 $\langle \text{proof} \rangle$

**lemma** *merge-length1-nonempty*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs \text{ } as \in \text{lists } (- \{\text{[]}\})$

**shows**  $us \in \text{lists } (- \{\text{[]}\})$

$\langle \text{proof} \rangle$

**lemma** *merge-length2-nonempty*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs \text{ } bs \in \text{lists } (- \{\text{[]}\})$

**shows**  $vs \in \text{lists } (- \{\text{[]}\})$

$\langle \text{proof} \rangle$

**lemma** *merge-length1-gt-0*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs \text{ } as \neq []$

**shows**  $\text{length } us > 0$

$\langle \text{proof} \rangle$

**lemma** *merge-length-le*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs$

**shows**  $\text{length } vs \leq \text{length } us$

$\langle \text{proof} \rangle$

**lemma** *merge-length-le-Suc*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs$

**shows**  $\text{length } us \leq \text{Suc } (\text{length } vs)$

$\langle \text{proof} \rangle$

**lemma** *merge-length-less2*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs$

**shows**  $\text{length } vs \leq \text{length } as$

$\langle \text{proof} \rangle$

**lemma** *merge-preserves*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs$

**shows**  $\text{concat } as = \text{concat } us \wedge \text{concat } bs = \text{concat } vs$

$\langle \text{proof} \rangle$

**lemma** *merge-interact*:

**assumes**  $\text{merge } as \text{ } bs \text{ } us \text{ } vs \text{ } \text{strict-sorted } (\text{concat } as) \text{ } \text{strict-sorted } (\text{concat } bs)$

$bs \in \text{lists } (- \{\square\})$   
**shows** *strict-sorted* (*interact us vs*)  
*<proof>*

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

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

**lemma** *length-hd-le-concat*:  
**assumes**  $as \neq []$  **shows**  $\text{length } (\text{hd } as) \leq \text{length } (\text{concat } as)$   
*<proof>*

**lemma** *length-hd-merge2*:  
**assumes** *merge as bs us vs*  
**shows**  $\text{length } (\text{hd } bs) \leq \text{length } (\text{hd } vs)$   
*<proof>*

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

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

**proposition** *merge-exists*:  
**assumes** *strict-sorted (concat as) strict-sorted (concat bs)*  
 $as \in \text{lists } (- \{\square\})$   $bs \in \text{lists } (- \{\square\})$   
 $\text{hd } as < \text{hd } bs$   $as \neq []$   $bs \neq []$   
**and** *disj*:  $\bigwedge a b. \llbracket a \in \text{list.set } as; b \in \text{list.set } bs \rrbracket \implies a < b \vee b < a$   
**shows**  $\exists us \text{ vs. merge } as \text{ bs } us \text{ vs}$   
*<proof>*

### 3.11.4 Actual proof of Larson's Lemma 3.8

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

$\exists l. \text{Form } l \ u \wedge (l > 0 \longrightarrow [\text{enum } N \ l] < \text{inter-scheme } l \ u \wedge \text{List.set}$   
 $(\text{inter-scheme } l \ u) \subseteq N)$   
 $\langle \text{proof} \rangle$

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

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

**corollary** *ordertype-eq-ordertype-iso-ll*:

**assumes**  $\text{Field } (\text{Restr } (\text{lenlex less-than}) \ A) = A \ \text{Field } (\text{Restr } (\text{lenlex less-than}) \ B) = B$

**shows**  $(\text{ordertype } A \ (\text{lenlex less-than}) = \text{ordertype } B \ (\text{lenlex less-than}))$

$\longleftrightarrow (\exists f. \text{iso-ll } A \ B \ f)$

$\langle \text{proof} \rangle$

**theorem** *partition- $\omega\omega$ -aux*:

**assumes**  $\alpha \in \text{elts } \omega$

**shows**  $\text{partn-lst } (\text{lenlex less-than}) \ WW \ [\omega \uparrow \omega, \alpha] \ 2 \ (\text{is partn-lst } ?R \ WW \ [\omega \uparrow \omega, \alpha] \ 2)$

$\langle \text{proof} \rangle$

Theorem 3.1 of Jean A. Larson, *ibid.*

**theorem** *partition- $\omega\omega$* :  $\alpha \in \text{elts } \omega \implies \text{partn-lst-VWF } (\omega \uparrow \omega) \ [\omega \uparrow \omega, \alpha] \ 2$

$\langle \text{proof} \rangle$

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-Argraki for assistance at tricky moments.

## References

- [1] P. Erdős and E. C. Milner. A theorem in the partition calculus. *Canadian Mathematical Bulletin*, 15(4):501–505, Dec. 1972.
- [2] P. Erdős 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  $\omega^\omega$ . *Annals of Mathematical Logic*, 6(2):129–145, Dec. 1973.