

# Zermelo Fraenkel Set Theory in Higher-Order Logic

Lawrence C. Paulson  
Computer Laboratory  
University of Cambridge

February 6, 2026

## Abstract

This entry is a new formalisation of ZFC set theory in Isabelle/HOL. It is logically equivalent to Obua's HOLZF [2]; the point is to have the closest possible integration with the rest of Isabelle/HOL, minimising the amount of new notations and exploiting type classes.

There is a type  $V$  of sets and a function  $elts :: V \Rightarrow V\ set$  mapping a set to its elements. Classes simply have type  $V\ set$ , and the predicate *small* identifies those classes that correspond to actual sets. Type classes connected with orders and lattices are used to minimise the amount of new notation for concepts such as the subset relation, union and intersection. Basic concepts are formalised: Cartesian products, disjoint sums, natural numbers, functions, etc.

More advanced set-theoretic concepts, such as transfinite induction, ordinals, cardinals and the transitive closure of a set, are also provided. The definition of addition and multiplication for general sets (not just ordinals) follows Kirby [1]. The development includes essential results about cardinal arithmetic. It also develops ordinal exponentiation, Cantor normal form and the concept of indecomposable ordinals. There are numerous results about order types.

The theory provides two type classes with the aim of facilitating developments that combine  $V$  with other Isabelle/HOL types: *embeddable*, the class of types that can be injected into  $V$  (including  $V$  itself as well as  $V^*V$ ,  $V\ list$ , etc.), and *small*, the class of types that correspond to some ZF set.

# Contents

<b>1</b>	<b>The ZF Axioms, Ordinals and Transfinite Recursion</b>	<b>4</b>
1.1	Syntax and axioms . . . . .	4
1.2	Type classes and other basic setup . . . . .	7
1.3	Successor function . . . . .	12
1.4	Ordinals . . . . .	13
1.4.1	Transitive sets . . . . .	13
1.4.2	Zero, successor, sups . . . . .	14
1.4.3	Induction, Linearity, etc. . . . .	15
1.4.4	The natural numbers . . . . .	16
1.4.5	Limit ordinals . . . . .	18
1.4.6	Properties of LEAST for ordinals . . . . .	19
1.5	Transfinite Recursion and the V-levels . . . . .	21
<b>2</b>	<b>Cartesian products, Disjoint Sums, Ranks, Cardinals</b>	<b>22</b>
2.1	Ordered Pairs . . . . .	22
2.2	Generalized Cartesian product . . . . .	24
2.3	Disjoint Sum . . . . .	25
2.3.1	Equivalences for the injections and an elimination rule	25
2.3.2	Injection and freeness equivalences, for rewriting . . .	26
2.3.3	Applications of disjoint sums and pairs: general union theorems for small sets . . . . .	27
2.4	Generalised function space and lambda . . . . .	27
2.5	Transitive closure of a set . . . . .	29
2.6	Rank of a set . . . . .	31
2.7	Cardinal Numbers . . . . .	33
2.7.1	Transitive Closure and VWO . . . . .	33
2.7.2	Relation VWF . . . . .	34
2.8	Order types . . . . .	35
2.8.1	<i>ordermap</i> preserves the orderings in both directions .	36
2.9	More advanced <i>ordertype</i> and <i>ordermap</i> results . . . . .	37
2.10	Cardinality of an arbitrary HOL set . . . . .	43
2.11	Cardinality of a set . . . . .	43
2.12	Cardinality of a set . . . . .	45
2.13	Transfinite recursion for definitions based on the three cases of ordinals . . . . .	46
2.14	Cardinal Addition . . . . .	46
2.14.1	Cardinal addition is commutative . . . . .	46
2.14.2	Cardinal addition is associative . . . . .	46
2.15	Cardinal multiplication . . . . .	47
2.15.1	Cardinal multiplication is commutative . . . . .	47
2.15.2	Cardinal multiplication is associative . . . . .	47
2.15.3	Cardinal multiplication distributes over addition . . .	48

2.16	Some inequalities for multiplication . . . . .	48
2.17	The finite cardinals . . . . .	48
2.18	Infinite cardinals . . . . .	49
2.19	Toward's Kunen's Corollary 10.13 (1) . . . . .	50
2.20	The Aleph-sequence . . . . .	51
2.21	The ordinal $\omega 1$ . . . . .	53
<b>3</b>	<b>Addition and Multiplication of Sets</b>	<b>53</b>
3.1	Generalised Addition . . . . .	54
3.1.1	Addition is a monoid . . . . .	54
3.1.2	Deeper properties of addition . . . . .	56
3.1.3	Cancellation / set subtraction . . . . .	58
3.2	Generalised Difference . . . . .	59
3.3	Generalised Multiplication . . . . .	60
3.3.1	Proposition 4.3 . . . . .	61
3.3.2	Proposition 4.4-5 . . . . .	62
3.3.3	Theorem 4.6 . . . . .	63
3.3.4	Theorem 4.7 . . . . .	63
3.4	Ordertype properties . . . . .	65
<b>4</b>	<b>Exponentiation of ordinals</b>	<b>65</b>
<b>5</b>	<b>Cantor Normal Form</b>	<b>68</b>
5.1	Cantor normal form . . . . .	69
5.2	Simplified Cantor normal form . . . . .	70
5.3	Indecomposable ordinals . . . . .	72
5.4	From ordinals to order types . . . . .	74
<b>6</b>	<b>Type Classes for ZFC</b>	<b>75</b>
6.1	The class of embeddable types . . . . .	75
6.2	The class of small types . . . . .	76
<b>7</b>	<b>ZF sets corresponding to <math>\mathbb{R}</math> and <math>\mathbb{C}</math> and the cardinality of the continuum</b>	<b>78</b>
7.1	Making the embedding from the type class explicit . . . . .	78
7.2	The cardinality of the continuum . . . . .	78
7.3	Countable and uncountable sets . . . . .	79
<b>8</b>	<b>Acknowledgements</b>	<b>80</b>

```

theory ZFC-Library
  imports HOL-Library.Countable-Set HOL-Library.Equipollence HOL-Cardinals.Cardinals

begin

Equipollence and Lists.

lemma countable-iff-lepoll: countable  $A \longleftrightarrow A \lesssim (UNIV :: nat\ set)$ 
  <proof>

lemma infinite-times-epoll-self:
  assumes infinite  $A$  shows  $A \times A \approx A$ 
  <proof>

lemma infinite-finite-times-lepoll-self:
  assumes infinite  $A$  finite  $B$  shows  $A \times B \lesssim A$ 
  <proof>

lemma lists-n-lepoll-self:
  assumes infinite  $A$  shows  $\{l \in lists\ A.\ length\ l = n\} \lesssim A$ 
  <proof>

lemma infinite-epoll-lists:
  assumes infinite  $A$  shows lists  $A \approx A$ 
  <proof>

end

```

## 1 The ZF Axioms, Ordinals and Transfinite Recursion

```

theory ZFC-in-HOL
  imports ZFC-Library

```

```

begin

```

### 1.1 Syntax and axioms

```

hide-const (open) list.set Sum subset

```

```

unbundle lattice-syntax

```

```

typedecl V

```

Presentation refined by Dmitriy Traytel

```

axiomatization elts ::  $V \Rightarrow V\ set$ 

```

```

where ext [intro?]:  $elts\ x = elts\ y \implies x=y$ 

```

```

and down-raw:  $Y \subseteq elts\ x \implies Y \in range\ elts$ 

```

```

and Union-raw:  $X \in range\ elts \implies Union\ (elts\ ` X) \in range\ elts$ 

```

**and** *Pow-raw*:  $X \in \text{range elts} \implies \text{inv elts } ' \text{Pow } X \in \text{range elts}$   
**and** *replacement-raw*:  $X \in \text{range elts} \implies f ' X \in \text{range elts}$   
**and** *inf-raw*:  $\text{range } (g :: \text{nat} \Rightarrow V) \in \text{range elts}$   
**and** *foundation*:  $\text{wf } \{(x,y). x \in \text{elts } y\}$

**lemma** *mem-not-refl* [*simp*]:  $i \notin \text{elts } i$   
 ⟨*proof*⟩

**lemma** *mem-not-sym*:  $\neg (x \in \text{elts } y \wedge y \in \text{elts } x)$   
 ⟨*proof*⟩

A set is small if it can be injected into the extension of a V-set.

**definition** *small* :: 'a set  $\Rightarrow$  bool  
**where** *small*  $X \equiv \exists V\text{-of} :: 'a \Rightarrow V. \text{inj-on } V\text{-of } X \wedge V\text{-of } ' X \in \text{range elts}$

**lemma** *small-empty* [*iff*]: *small* {}  
 ⟨*proof*⟩

**lemma** *small-iff-range*: *small*  $X \longleftrightarrow X \in \text{range elts}$   
 ⟨*proof*⟩

**lemma** *small-epoll*: *small*  $A \longleftrightarrow (\exists x. \text{elts } x \approx A)$   
 ⟨*proof*⟩

Small classes can be mapped to sets.

**definition** *set* ::  $V \text{ set} \Rightarrow V$   
**where** *set*  $X \equiv (\text{if } \text{small } X \text{ then } \text{inv elts } X \text{ else } \text{inv elts } \{\})$

**lemma** *set-of-elts* [*simp*]: *set* (elts  $x$ ) =  $x$   
 ⟨*proof*⟩

**lemma** *elts-of-set* [*simp*]: elts (set  $X$ ) = (if *small*  $X$  then  $X$  else {})  
 ⟨*proof*⟩

**lemma** *down*:  $Y \subseteq \text{elts } x \implies \text{small } Y$   
 ⟨*proof*⟩

**lemma** *Union* [*intro*]: *small*  $X \implies \text{small } (\text{Union } (\text{elts } ' X))$   
 ⟨*proof*⟩

**lemma** *Pow*: *small*  $X \implies \text{small } (\text{set } ' \text{Pow } X)$   
 ⟨*proof*⟩

**declare** *replacement-raw* [*intro,simp*]

**lemma** *replacement* [*intro,simp*]:  
**assumes** *small*  $X$   
**shows** *small* ( $f ' X$ )  
 ⟨*proof*⟩

**lemma** *small-image-iff* [*simp*]:  $\text{inj-on } f \ A \implies \text{small } (f \ ' \ A) \longleftrightarrow \text{small } A$   
 ⟨*proof*⟩

A little bootstrapping is needed to characterise *small* for sets of arbitrary type.

**lemma** *inf*:  $\text{small } (\text{range } (g :: \text{nat} \Rightarrow V))$   
 ⟨*proof*⟩

**lemma** *small-image-nat-V* [*simp*]:  $\text{small } (g \ ' \ N)$  **for**  $g :: \text{nat} \Rightarrow V$   
 ⟨*proof*⟩

**lemma** *Finite-V*:  
**fixes**  $X :: V \ \text{set}$   
**assumes** *finite*  $X$  **shows** *small*  $X$   
 ⟨*proof*⟩

**lemma** *small-insert-V*:  
**fixes**  $X :: V \ \text{set}$   
**assumes** *small*  $X$   
**shows** *small* (*insert*  $a \ X$ )  
 ⟨*proof*⟩

**lemma** *small-UN-V* [*simp,intro*]:  
**fixes**  $B :: 'a \Rightarrow V \ \text{set}$   
**assumes**  $X: \text{small } X$  **and**  $B: \bigwedge x. x \in X \implies \text{small } (B \ x)$   
**shows** *small* ( $\bigcup_{x \in X}. B \ x$ )  
 ⟨*proof*⟩

**definition** *vinsert* **where**  $\text{vinsert } x \ y \equiv \text{set } (\text{insert } x \ (\text{elts } y))$

**lemma** *elts-vinsert* [*simp*]:  $\text{elts } (\text{vinsert } x \ y) = \text{insert } x \ (\text{elts } y)$   
 ⟨*proof*⟩

**definition** *succ* **where**  $\text{succ } x \equiv \text{vinsert } x \ x$

**lemma** *elts-succ* [*simp*]:  $\text{elts } (\text{succ } x) = \text{insert } x \ (\text{elts } x)$   
 ⟨*proof*⟩

**lemma** *finite-imp-small*:  
**assumes** *finite*  $X$  **shows** *small*  $X$   
 ⟨*proof*⟩

**lemma** *small-insert*:  
**assumes** *small*  $X$   
**shows** *small* (*insert*  $a \ X$ )  
 ⟨*proof*⟩

**lemma** *smaller-than-small*:

**assumes** *small*  $A \subseteq B$  **shows** *small*  $B$   
 ⟨*proof*⟩

**lemma** *small-insert-iff* [*iff*]: *small* (*insert*  $a$   $X$ )  $\longleftrightarrow$  *small*  $X$   
 ⟨*proof*⟩

**lemma** *small-iff*: *small*  $X \longleftrightarrow (\exists x. X = \text{elts } x)$   
 ⟨*proof*⟩

**lemma** *small-elts* [*iff*]: *small* (*elts*  $x$ )  
 ⟨*proof*⟩

**lemma** *small-diff* [*iff*]: *small* (*elts*  $a - X$ )  
 ⟨*proof*⟩

**lemma** *small-set* [*simp*]: *small* (*list.set*  $xs$ )  
 ⟨*proof*⟩

**lemma** *small-upair*: *small*  $\{x,y\}$   
 ⟨*proof*⟩

**lemma** *small-Un-elts*: *small* (*elts*  $x \cup \text{elts } y$ )  
 ⟨*proof*⟩

**lemma** *small-eqcong*:  $\llbracket \text{small } X; X \approx Y \rrbracket \implies \text{small } Y$   
 ⟨*proof*⟩

**lemma** *lepoll-small*:  $\llbracket \text{small } Y; X \lesssim Y \rrbracket \implies \text{small } X$   
 ⟨*proof*⟩

**lemma** *big-UNIV* [*simp*]:  $\neg \text{small } (UNIV::V \text{ set})$  (**is**  $\neg \text{small } ?U$ )  
 ⟨*proof*⟩

**lemma** *inj-on-set*: *inj-on set* (*Collect small*)  
 ⟨*proof*⟩

**lemma** *set-injective* [*simp*]:  $\llbracket \text{small } X; \text{small } Y \rrbracket \implies \text{set } X = \text{set } Y \longleftrightarrow X=Y$   
 ⟨*proof*⟩

## 1.2 Type classes and other basic setup

**instantiation**  $V :: \text{zero}$   
**begin**  
**definition** *zero-V* **where**  $0 \equiv \text{set } \{\}$   
**instance** ⟨*proof*⟩  
**end**

**lemma** *elts-0* [*simp*]: *elts*  $0 = \{\}$   
 ⟨*proof*⟩

**lemma** *set-empty* [*simp*]:  $set \{\} = 0$   
⟨*proof*⟩

**instantiation**  $V :: one$   
**begin**  
**definition** *one-V* **where**  $1 \equiv succ\ 0$   
**instance** ⟨*proof*⟩  
**end**

**lemma** *elts-1* [*simp*]:  $elts\ 1 = \{0\}$   
⟨*proof*⟩

**lemma** *insert-neq-0* [*simp*]:  $set\ (insert\ a\ X) = 0 \longleftrightarrow \neg\ small\ X$   
⟨*proof*⟩

**lemma** *elts-eq-empty-iff* [*simp*]:  $elts\ x = \{\} \longleftrightarrow x = 0$   
⟨*proof*⟩

**instantiation**  $V :: distrib-lattice$   
**begin**

**definition** *inf-V* **where**  $inf-V\ x\ y \equiv set\ (elts\ x \cap elts\ y)$

**definition** *sup-V* **where**  $sup-V\ x\ y \equiv set\ (elts\ x \cup elts\ y)$

**definition** *less-eq-V* **where**  $less-eq-V\ x\ y \equiv elts\ x \subseteq elts\ y$

**definition** *less-V* **where**  $less-V\ x\ y \equiv less-eq\ x\ y \wedge x \neq (y::V)$

**instance**  
⟨*proof*⟩  
**end**

**lemma** *V-equalityI* [*intro*]:  $(\bigwedge x. x \in elts\ a \longleftrightarrow x \in elts\ b) \Longrightarrow a = b$   
⟨*proof*⟩

**lemma** *vsubsetI* [*intro!*]:  $(\bigwedge x. x \in elts\ a \Longrightarrow x \in elts\ b) \Longrightarrow a \leq b$   
⟨*proof*⟩

**lemma** *vsubsetD* [*elim, intro?*]:  $a \leq b \Longrightarrow c \in elts\ a \Longrightarrow c \in elts\ b$   
⟨*proof*⟩

**lemma** *rev-vsubsetD*:  $c \in elts\ a \Longrightarrow a \leq b \Longrightarrow c \in elts\ b$   
— The same, with reversed premises for use with *erule* – cf.  $\llbracket ?P; ?P \longrightarrow ?Q \rrbracket \Longrightarrow ?Q$ .  
⟨*proof*⟩

**lemma** *vsubsetCE* [*elim, no-atp*]:  $a \leq b \Longrightarrow (c \notin elts\ a \Longrightarrow P) \Longrightarrow (c \in elts\ b \Longrightarrow$

$P) \implies P$

— Classical elimination rule.

$\langle proof \rangle$

**lemma** *set-image-le-iff*:  $small\ A \implies set\ (f\ 'A) \leq B \longleftrightarrow (\forall x \in A. f\ x \in elts\ B)$

$\langle proof \rangle$

**lemma** *eq0-iff*:  $x = 0 \longleftrightarrow (\forall y. y \notin elts\ x)$

$\langle proof \rangle$

**lemma** *less-eq-V-0-iff* [*simp*]:  $x \leq 0 \longleftrightarrow x = 0$  **for**  $x :: V$

$\langle proof \rangle$

**lemma** *subset-iff-less-eq-V*:

**assumes** *small B* **shows**  $A \subseteq B \longleftrightarrow set\ A \leq set\ B \wedge small\ A$

$\langle proof \rangle$

**lemma** *small-Collect* [*simp*]:  $small\ A \implies small\ \{x \in A. P\ x\}$

$\langle proof \rangle$

**lemma** *small-Union-iff*:  $small\ (\bigcup (elts\ 'X)) \longleftrightarrow small\ X$

$\langle proof \rangle$

**lemma** *not-less-0* [*iff*]:

**fixes**  $x :: V$  **shows**  $\neg x < 0$

$\langle proof \rangle$

**lemma** *le-0* [*iff*]:

**fixes**  $x :: V$  **shows**  $0 \leq x$

$\langle proof \rangle$

**lemma** *min-0L* [*simp*]:  $min\ 0\ n = 0$  **for**  $n :: V$

$\langle proof \rangle$

**lemma** *min-0R* [*simp*]:  $min\ n\ 0 = 0$  **for**  $n :: V$

$\langle proof \rangle$

**lemma** *neq0-conv*:  $\bigwedge n :: V. n \neq 0 \longleftrightarrow 0 < n$

$\langle proof \rangle$

**definition**  $VPow :: V \Rightarrow V$

**where**  $VPow\ x \equiv set\ (set\ 'Pow\ (elts\ x))$

**lemma** *VPow-iff* [*iff*]:  $y \in elts\ (VPow\ x) \longleftrightarrow y \leq x$

$\langle proof \rangle$

**lemma** *VPow-le-VPow-iff* [*simp*]:  $VPow\ a \leq VPow\ b \longleftrightarrow a \leq b$

$\langle proof \rangle$

**lemma** *elts-VPow*:  $elts (VPow\ x) = set\ 'Pow\ (elts\ x)$   
⟨*proof*⟩

**lemma** *small-sup-iff* [*simp*]:  $small\ (X\ \cup\ Y) \longleftrightarrow small\ X \wedge small\ Y$  **for**  $X::V\ set$   
⟨*proof*⟩

**lemma** *elts-sup-iff* [*simp*]:  $elts\ (x\ \sqcup\ y) = elts\ x \cup elts\ y$   
⟨*proof*⟩

**lemma** *trad-foundation*:  
**assumes**  $z: z \neq 0$  **shows**  $\exists w. w \in elts\ z \wedge w \sqcap z = 0$   
⟨*proof*⟩

**instantiation**  $V :: Sup$   
**begin**  
**definition** *Sup-V* **where**  $Sup-V\ X \equiv if\ small\ X\ then\ set\ (Union\ (elts\ 'X))\ else\ 0$   
**instance** ⟨*proof*⟩  
**end**

**instantiation**  $V :: Inf$   
**begin**  
**definition** *Inf-V* **where**  $Inf-V\ X \equiv if\ X = \{\}\ then\ 0\ else\ set\ (Inter\ (elts\ 'X))$   
**instance** ⟨*proof*⟩  
**end**

**lemma** *V-disjoint-iff*:  $x \sqcap y = 0 \longleftrightarrow elts\ x \cap elts\ y = \{\}$   
⟨*proof*⟩

I've no idea why *bdd-above* is treated differently from *bdd-below*, but anyway

**lemma** *bdd-above-iff-small* [*simp*]:  $bdd-above\ X = small\ X$  **for**  $X::V\ set$   
⟨*proof*⟩

**instantiation**  $V :: conditionally-complete-lattice$   
**begin**

**definition** *bdd-below-V* **where**  $bdd-below-V\ X \equiv X \neq \{\}$

**instance**  
⟨*proof*⟩  
**end**

**lemma** *Sup-upper*:  $\llbracket x \in A; small\ A \rrbracket \implies x \leq \bigsqcup A$  **for**  $A::V\ set$   
⟨*proof*⟩

**lemma** *Sup-least*:  
**fixes**  $z::V$  **shows**  $(\bigwedge x. x \in A \implies x \leq z) \implies \bigsqcup A \leq z$

*<proof>*

**lemma** *Sup-empty* [simp]:  $\bigsqcup \{\} = (0::V)$   
*<proof>*

**lemma** *elts-Sup* [simp]: *small*  $X \implies \text{elts } (\bigsqcup X) = \bigcup (\text{elts } ' X)$   
*<proof>*

**lemma** *sup-V-0-left* [simp]:  $0 \sqcup a = a$  **and** *sup-V-0-right* [simp]:  $a \sqcup 0 = a$  **for**  $a::V$   
*<proof>*

**lemma** *Sup-V-insert*:  
**fixes**  $x::V$  **assumes** *small*  $A$  **shows**  $\bigsqcup (\text{insert } x A) = x \sqcup \bigsqcup A$   
*<proof>*

**lemma** *Sup-Un-distrib*:  $\llbracket \text{small } A; \text{small } B \rrbracket \implies \bigsqcup (A \cup B) = \bigsqcup A \sqcup \bigsqcup B$  **for**  $A::V$  *set*  
*<proof>*

**lemma** *SUP-sup-distrib*:  
**fixes**  $f::V \Rightarrow V$   
**shows** *small*  $A \implies (\bigsqcup x \in A. f x \sqcup g x) = \bigsqcup (f ' A) \sqcup \bigsqcup (g ' A)$   
*<proof>*

**lemma** *SUP-const* [simp]:  $(\bigsqcup y \in A. a) = (\text{if } A = \{\} \text{ then } (0::V) \text{ else } a)$   
*<proof>*

**lemma** *cSUP-subset-mono*:  
**fixes**  $f::'a \Rightarrow V$  *set* **and**  $g::'a \Rightarrow V$  *set*  
**shows**  $\llbracket A \subseteq B; \bigwedge x. x \in A \implies f x \leq g x \rrbracket \implies \bigsqcup (f ' A) \leq \bigsqcup (g ' B)$   
*<proof>*

**lemma** *mem-Sup-iff* [iff]:  $x \in \text{elts } (\bigsqcup X) \longleftrightarrow x \in \bigcup (\text{elts } ' X) \wedge \text{small } X$   
*<proof>*

**lemma** *cSUP-UNION*:  
**fixes**  $B::V \Rightarrow V$  *set* **and**  $f::V \Rightarrow V$   
**assumes** *ne*: *small*  $A$  **and** *bdd-UN*: *small*  $(\bigcup x \in A. f ' B x)$   
**shows**  $\bigsqcup (f ' (\bigcup x \in A. B x)) = \bigsqcup ((\lambda x. \bigsqcup (f ' B x)) ' A)$   
*<proof>*

**lemma** *Sup-subset-mono*: *small*  $B \implies A \subseteq B \implies \text{Sup } A \leq \text{Sup } B$  **for**  $A::V$  *set*  
*<proof>*

**lemma** *Sup-le-iff*: *small*  $A \implies \text{Sup } A \leq a \longleftrightarrow (\forall x \in A. x \leq a)$  **for**  $A::V$  *set*  
*<proof>*

**lemma** *SUP-le-iff*: *small*  $(f ' A) \implies \bigsqcup (f ' A) \leq u \longleftrightarrow (\forall x \in A. f x \leq u)$  **for**  $f::$

$V \Rightarrow V$   
*<proof>*

**lemma** *Sup-eq-0-iff* [simp]:  $\sqcup A = 0 \iff A \subseteq \{0\} \vee \neg \text{small } A$  **for**  $A :: V \text{ set}$   
*<proof>*

**lemma** *Sup-Union-commute*:  
**fixes**  $f :: V \Rightarrow V \text{ set}$   
**assumes**  $\text{small } A \wedge x. x \in A \implies \text{small } (f x)$   
**shows**  $\sqcup (\bigcup x \in A. f x) = (\sqcup x \in A. \sqcup (f x))$   
*<proof>*

**lemma** *Sup-eq-Sup*:  
**fixes**  $B :: V \text{ set}$   
**assumes**  $B \subseteq A$  **small**  $A$  **and**  $*$ :  $\bigwedge x. x \in A \implies \exists y \in B. x \leq y$   
**shows**  $\text{Sup } A = \text{Sup } B$   
*<proof>*

### 1.3 Successor function

**lemma** *vinsert-not-empty* [simp]:  $\text{vinsert } a \ A \neq 0$   
**and** *empty-not-vinsert* [simp]:  $0 \neq \text{vinsert } a \ A$   
*<proof>*

**lemma** *succ-not-0* [simp]:  $\text{succ } n \neq 0$  **and** *zero-not-succ* [simp]:  $0 \neq \text{succ } n$   
*<proof>*

**instantiation**  $V :: \text{zero-neq-one}$   
**begin**  
**instance**  
*<proof>*  
**end**

**instantiation**  $V :: \text{zero-less-one}$   
**begin**  
**instance**  
*<proof>*  
**end**

**lemma** *succ-ne-self* [simp]:  $i \neq \text{succ } i$   
*<proof>*

**lemma** *succ-notin-self*:  $\text{succ } i \notin \text{elts } i$   
*<proof>*

**lemma** *le-succE*:  $\text{succ } i \leq \text{succ } j \implies i \leq j$   
*<proof>*

**lemma** *succ-inject-iff* [iff]:  $\text{succ } i = \text{succ } j \iff i = j$

*<proof>*

**lemma** *inj-succ*: *inj succ*

*<proof>*

**lemma** *succ-neq-zero*: *succ x ≠ 0*

*<proof>*

**definition** *pred* **where** *pred i ≡ THE j. i = succ j*

**lemma** *pred-succ* [*simp*]: *pred (succ i) = i*

*<proof>*

## 1.4 Ordinals

**definition** *Transset* **where** *Transset x ≡ ∀ y ∈ elts x. y ≤ x*

**definition** *Ord* **where** *Ord x ≡ Transset x ∧ (∀ y ∈ elts x. Transset y)*

**abbreviation** *ON* **where** *ON ≡ Collect Ord*

### 1.4.1 Transitive sets

**lemma** *Transset-0* [*iff*]: *Transset 0*

*<proof>*

**lemma** *Transset-succ* [*intro*]:

**assumes** *Transset x* **shows** *Transset (succ x)*

*<proof>*

**lemma** *Transset-Sup*:

**assumes**  $\bigwedge x. x \in X \implies \text{Transset } x$  **shows** *Transset ( $\bigsqcup X$ )*

*<proof>*

**lemma** *Transset-sup*:

**assumes** *Transset x* *Transset y* **shows** *Transset (x  $\sqcup$  y)*

*<proof>*

**lemma** *Transset-inf*:  $\llbracket \text{Transset } i; \text{Transset } j \rrbracket \implies \text{Transset } (i \sqcap j)$

*<proof>*

**lemma** *Transset-VPow*: *Transset(i)  $\implies$  Transset(VPow(i))*

*<proof>*

**lemma** *Transset-Inf*:  $(\bigwedge i. i \in A \implies \text{Transset } i) \implies \text{Transset } (\bigsqcap A)$

*<proof>*

**lemma** *Transset-SUP*:  $(\bigwedge x. x \in A \implies \text{Transset } (B x)) \implies \text{Transset } (\bigsqcup (B \text{ ` } A))$

*<proof>*

**lemma** *Transset-INT*:  $(\bigwedge x. x \in A \implies \text{Transset } (B x)) \implies \text{Transset } (\prod (B ' A))$   
*<proof>*

### 1.4.2 Zero, successor, sups

**lemma** *Ord-0 [iff]*: *Ord 0*  
*<proof>*

**lemma** *Ord-succ [intro]*:  
**assumes** *Ord x* **shows** *Ord (succ x)*  
*<proof>*

**lemma** *Ord-Sup*:  
**assumes**  $\bigwedge x. x \in X \implies \text{Ord } x$  **shows** *Ord ( $\sqcup X$ )*  
*<proof>*

**lemma** *Ord-Union*:  
**assumes**  $\bigwedge x. x \in X \implies \text{Ord } x$  *small X* **shows** *Ord (set ( $\bigcup$  (elts ' X)))*  
*<proof>*

**lemma** *Ord-sup*:  
**assumes** *Ord x Ord y* **shows** *Ord (x  $\sqcup$  y)*  
*<proof>*

**lemma** *big-ON [simp]*:  $\neg$  *small ON*  
*<proof>*

**lemma** *Ord-1 [iff]*: *Ord 1*  
*<proof>*

**lemma** *OrdmemD*: *Ord k*  $\implies j \in \text{elts } k \implies j < k$   
*<proof>*

**lemma** *Ord-trans*:  $\llbracket i \in \text{elts } j; j \in \text{elts } k; \text{Ord } k \rrbracket \implies i \in \text{elts } k$   
*<proof>*

**lemma** *mem-0-Ord*:  
**assumes** *k: Ord k* **and** *knz: k  $\neq$  0* **shows** *0  $\in$  elts k*  
*<proof>*

**lemma** *Ord-in-Ord*:  $\llbracket \text{Ord } k; m \in \text{elts } k \rrbracket \implies \text{Ord } m$   
*<proof>*

**lemma** *OrdI*:  $\llbracket \text{Transset } i; \bigwedge x. x \in \text{elts } i \implies \text{Transset } x \rrbracket \implies \text{Ord } i$   
*<proof>*

**lemma** *Ord-is-Transset*: *Ord i*  $\implies \text{Transset } i$   
*<proof>*

**lemma** *Ord-contains-Transset*:  $\llbracket \text{Ord } i; j \in \text{elts } i \rrbracket \implies \text{Transset } j$   
 ⟨proof⟩

**lemma** *ON-imp-Ord*:  
**assumes**  $H \subseteq \text{ON } x \in H$   
**shows**  $\text{Ord } x$   
 ⟨proof⟩

**lemma** *elts-subset-ON*:  $\text{Ord } \alpha \implies \text{elts } \alpha \subseteq \text{ON}$   
 ⟨proof⟩

**lemma** *Transset-pred [simp]*:  $\text{Transset } x \implies \sqcup (\text{elts } (\text{succ } x)) = x$   
 ⟨proof⟩

**lemma** *Ord-pred [simp]*:  $\text{Ord } \beta \implies \sqcup (\text{insert } \beta (\text{elts } \beta)) = \beta$   
 ⟨proof⟩

### 1.4.3 Induction, Linearity, etc.

**lemma** *Ord-induct [consumes 1, case-names step]*:  
**assumes**  $k: \text{Ord } k$   
**and step**:  $\bigwedge x. \llbracket \text{Ord } x; \bigwedge y. y \in \text{elts } x \implies P y \rrbracket \implies P x$   
**shows**  $P k$   
 ⟨proof⟩

Comparability of ordinals

**lemma** *Ord-linear*:  $\text{Ord } k \implies \text{Ord } l \implies k \in \text{elts } l \vee k=l \vee l \in \text{elts } k$   
 ⟨proof⟩

The trichotomy law for ordinals

**lemma** *Ord-linear-lt*:  
**assumes**  $\text{Ord } k \text{ Ord } l$   
**obtains**  $(lt) k < l \mid (eq) k=l \mid (gt) l < k$   
 ⟨proof⟩

**lemma** *Ord-linear2*:  
**assumes**  $\text{Ord } k \text{ Ord } l$   
**obtains**  $(lt) k < l \mid (ge) l \leq k$   
 ⟨proof⟩

**lemma** *Ord-linear-le*:  
**assumes**  $\text{Ord } k \text{ Ord } l$   
**obtains**  $(le) k \leq l \mid (ge) l \leq k$   
 ⟨proof⟩

**lemma** *union-less-iff [simp]*:  $\llbracket \text{Ord } i; \text{Ord } j \rrbracket \implies i \sqcup j < k \iff i < k \wedge j < k$   
 ⟨proof⟩

**lemma** *Ord-mem-iff-lt*:  $\text{Ord } k \implies \text{Ord } l \implies k \in \text{elts } l \iff k < l$

*<proof>*

**lemma** *Ord-Collect-lt*:  $\text{Ord } \alpha \implies \{\xi. \text{Ord } \xi \wedge \xi < \alpha\} = \text{elts } \alpha$   
*<proof>*

**lemma** *Ord-not-less*:  $\llbracket \text{Ord } x; \text{Ord } y \rrbracket \implies \neg x < y \longleftrightarrow y \leq x$   
*<proof>*

**lemma** *Ord-not-le*:  $\llbracket \text{Ord } x; \text{Ord } y \rrbracket \implies \neg x \leq y \longleftrightarrow y < x$   
*<proof>*

**lemma** *le-succ-iff*:  $\text{Ord } i \implies \text{Ord } j \implies \text{succ } i \leq \text{succ } j \longleftrightarrow i \leq j$   
*<proof>*

**lemma** *succ-le-iff*:  $\text{Ord } i \implies \text{Ord } j \implies \text{succ } i \leq j \longleftrightarrow i < j$   
*<proof>*

**lemma** *succ-in-Sup-Ord*:  
**assumes** *eq*:  $\text{succ } \beta = \bigsqcup A$  **and** *small A*  $A \subseteq \text{ON}$  *Ord*  $\beta$   
**shows**  $\text{succ } \beta \in A$   
*<proof>*

**lemma** *in-succ-iff*:  $\text{Ord } i \implies j \in \text{elts } (\text{ZFC-in-HOL.succ } i) \longleftrightarrow \text{Ord } j \wedge j \leq i$   
*<proof>*

**lemma** *zero-in-succ* [*simp,intro*]:  $\text{Ord } i \implies 0 \in \text{elts } (\text{succ } i)$   
*<proof>*

**lemma** *less-succ-self*:  $x < \text{succ } x$   
*<proof>*

**lemma** *Ord-finite-Sup*:  $\llbracket \text{finite } A; A \subseteq \text{ON}; A \neq \{\} \rrbracket \implies \bigsqcup A \in A$   
*<proof>*

#### 1.4.4 The natural numbers

**primrec** *ord-of-nat* ::  $\text{nat} \Rightarrow V$  **where**  
*ord-of-nat*  $0 = 0$   
| *ord-of-nat* (*Suc*  $n$ ) =  $\text{succ } (\text{ord-of-nat } n)$

**lemma** *ord-of-nat-eq-initial*:  $\text{ord-of-nat } n = \text{set } (\text{ord-of-nat } \{..<n\})$   
*<proof>*

**lemma** *mem-ord-of-nat-iff* [*simp*]:  $x \in \text{elts } (\text{ord-of-nat } n) \longleftrightarrow (\exists m < n. x = \text{ord-of-nat } m)$   
*<proof>*

**lemma** *elts-ord-of-nat*:  $\text{elts } (\text{ord-of-nat } k) = \text{ord-of-nat } \{..<k\}$   
*<proof>*

**lemma** *Ord-equality*:  $\text{Ord } i \implies i = \bigsqcup (\text{succ } \cdot \text{elts } i)$   
*<proof>*

**lemma** *Ord-ord-of-nat [simp]*:  $\text{Ord } (\text{ord-of-nat } k)$   
*<proof>*

**lemma** *ord-of-nat-equality*:  $\text{ord-of-nat } n = \bigsqcup ((\text{succ } \circ \text{ord-of-nat}) \cdot \{..<n\})$   
*<proof>*

**definition**  $\omega :: V$  **where**  $\omega \equiv \text{set } (\text{range } \text{ord-of-nat})$

**lemma** *elts- $\omega$* :  $\text{elts } \omega = \{\alpha. \exists n. \alpha = \text{ord-of-nat } n\}$   
*<proof>*

**lemma** *nat-into-Ord [simp]*:  $n \in \text{elts } \omega \implies \text{Ord } n$   
*<proof>*

**lemma** *Sup- $\omega$* :  $\bigsqcup (\text{elts } \omega) = \omega$   
*<proof>*

**lemma** *Ord- $\omega$  [iff]*:  $\text{Ord } \omega$   
*<proof>*

**lemma** *zero-in-omega [iff]*:  $0 \in \text{elts } \omega$   
*<proof>*

**lemma** *succ-in-omega [simp]*:  $n \in \text{elts } \omega \implies \text{succ } n \in \text{elts } \omega$   
*<proof>*

**lemma** *ord-of-eq-0*:  $\text{ord-of-nat } j = 0 \implies j = 0$   
*<proof>*

**lemma** *ord-of-nat-le-omega*:  $\text{ord-of-nat } n \leq \omega$   
*<proof>*

**lemma** *ord-of-eq-0-iff [simp]*:  $\text{ord-of-nat } n = 0 \iff n=0$   
*<proof>*

**lemma** *ord-of-nat-inject [iff]*:  $\text{ord-of-nat } i = \text{ord-of-nat } j \iff i=j$   
*<proof>*

**corollary** *inj-ord-of-nat*:  $\text{inj } \text{ord-of-nat}$   
*<proof>*

**corollary** *countable*:  
**assumes** *countable X shows small X*  
*<proof>*

**corollary** *infinite- $\omega$* : *infinite* (elts  $\omega$ )  
(proof)

**corollary** *ord-of-nat-mono-iff* [iff]: *ord-of-nat*  $i \leq$  *ord-of-nat*  $j \iff i \leq j$   
(proof)

**corollary** *ord-of-nat-strict-mono-iff* [iff]: *ord-of-nat*  $i <$  *ord-of-nat*  $j \iff i < j$   
(proof)

**lemma** *small-image-nat* [simp]:  
fixes  $N :: \text{nat set}$  shows *small* ( $g \text{ ' } N$ )  
(proof)

**lemma** *finite-Ord-omega*:  $\alpha \in \text{elts } \omega \implies \text{finite} (\text{elts } \alpha)$   
(proof)

**lemma** *infinite-Ord-omega*:  $\text{Ord } \alpha \implies \text{infinite} (\text{elts } \alpha) \implies \omega \leq \alpha$   
(proof)

**lemma** *ord-of-minus-1*:  $n > 0 \implies \text{ord-of-nat } n = \text{succ} (\text{ord-of-nat } (n - 1))$   
(proof)

**lemma** *card-ord-of-nat* [simp]:  $\text{card} (\text{elts} (\text{ord-of-nat } m)) = m$   
(proof)

**lemma** *ord-of-nat- $\omega$*  [iff]: *ord-of-nat*  $n \in \text{elts } \omega$   
(proof)

**lemma** *succ- $\omega$ -iff* [iff]:  $\text{succ } n \in \text{elts } \omega \iff n \in \text{elts } \omega$   
(proof)

**lemma**  *$\omega$ -gt0* [simp]:  $\omega > 0$   
(proof)

**lemma**  *$\omega$ -gt1* [simp]:  $\omega > 1$   
(proof)

### 1.4.5 Limit ordinals

**definition** *Limit* ::  $V \Rightarrow \text{bool}$   
where *Limit*  $i \equiv \text{Ord } i \wedge 0 \in \text{elts } i \wedge (\forall y. y \in \text{elts } i \longrightarrow \text{succ } y \in \text{elts } i)$

**lemma** *zero-not-Limit* [iff]:  $\neg \text{Limit } 0$   
(proof)

**lemma** *not-succ-Limit* [simp]:  $\neg \text{Limit}(\text{succ } i)$   
(proof)

**lemma** *Limit-is-Ord*:  $\text{Limit } \xi \implies \text{Ord } \xi$

*<proof>*

**lemma** *succ-in-Limit-iff*:  $Limit\ \xi \implies succ\ \alpha \in elts\ \xi \longleftrightarrow \alpha \in elts\ \xi$   
*<proof>*

**lemma** *Limit-eq-Sup-self* [*simp*]:  $Limit\ i \implies Sup\ (elts\ i) = i$   
*<proof>*

**lemma** *zero-less-Limit*:  $Limit\ \beta \implies 0 < \beta$   
*<proof>*

**lemma** *non-Limit-ord-of-nat* [*iff*]:  $\neg Limit\ (ord\text{-of-nat}\ m)$   
*<proof>*

**lemma** *Limit-omega* [*iff*]:  $Limit\ \omega$   
*<proof>*

**lemma** *omega-nonzero* [*simp*]:  $\omega \neq 0$   
*<proof>*

**lemma** *Ord-cases-lemma*:  
**assumes** *Ord* *k* **shows**  $k = 0 \vee (\exists j. k = succ\ j) \vee Limit\ k$   
*<proof>*

**lemma** *Ord-cases* [*cases type: V, case-names 0 succ limit*]:  
**assumes** *Ord* *k*  
**obtains**  $k = 0 \mid l$  **where** *Ord* *l*  $succ\ l = k \mid Limit\ k$   
*<proof>*

**lemma** *non-succ-LimitI*:  
**assumes**  $i \neq 0$  *Ord*(*i*)  $\wedge y. succ(y) \neq i$   
**shows** *Limit*(*i*)  
*<proof>*

**lemma** *Ord-induct3* [*consumes 1, case-names 0 succ Limit, induct type: V*]:  
**assumes**  $\alpha: Ord\ \alpha$   
**and**  $P\ 0 \wedge \alpha. \llbracket Ord\ \alpha; P\ \alpha \rrbracket \implies P\ (succ\ \alpha)$   
 $\wedge \alpha. \llbracket Limit\ \alpha; \wedge \xi. \xi \in elts\ \alpha \implies P\ \xi \rrbracket \implies P\ (\bigsqcup \xi \in elts\ \alpha. \xi)$   
**shows**  $P\ \alpha$   
*<proof>*

#### 1.4.6 Properties of LEAST for ordinals

**lemma**  
**assumes** *Ord* *k* *P* *k*  
**shows** *Ord-LeastI*:  $P\ (LEAST\ i. Ord\ i \wedge P\ i)$  **and** *Ord-Least-le*:  $(LEAST\ i. Ord\ i \wedge P\ i) \leq k$   
*<proof>*

**lemma** *Ord-Least*:

**assumes**  $Ord\ k\ P\ k$

**shows**  $Ord\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

**lemma** *Ord-LeastI-ex*:  $\exists i.\ Ord\ i\ \wedge\ P\ i \implies P\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

**lemma** *Ord-LeastI2*:

$\llbracket Ord\ a; P\ a; \bigwedge x.\ \llbracket Ord\ x; P\ x \rrbracket \implies Q\ x \rrbracket \implies Q\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

**lemma** *Ord-LeastI2-ex*:

$\exists a.\ Ord\ a\ \wedge\ P\ a \implies (\bigwedge x.\ \llbracket Ord\ x; P\ x \rrbracket \implies Q\ x) \implies Q\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

*<proof>*

**lemma** *Ord-LeastI2-wellorder*:

**assumes**  $Ord\ a\ P\ a$

**and**  $\bigwedge a.\ \llbracket P\ a; \forall b.\ Ord\ b\ \wedge\ P\ b \longrightarrow a \leq b \rrbracket \implies Q\ a$

**shows**  $Q\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

**lemma** *Ord-LeastI2-wellorder-ex*:

**assumes**  $\exists x.\ Ord\ x\ \wedge\ P\ x$

**and**  $\bigwedge a.\ \llbracket P\ a; \forall b.\ Ord\ b\ \wedge\ P\ b \longrightarrow a \leq b \rrbracket \implies Q\ a$

**shows**  $Q\ (LEAST\ i.\ Ord\ i\ \wedge\ P\ i)$

*<proof>*

**lemma** *not-less-Ord-Least*:  $\llbracket k < (LEAST\ x.\ Ord\ x\ \wedge\ P\ x); Ord\ k \rrbracket \implies \neg P\ k$

*<proof>*

**lemma** *exists-Ord-Least-iff*:  $(\exists \alpha.\ Ord\ \alpha\ \wedge\ P\ \alpha) \longleftrightarrow (\exists \alpha.\ Ord\ \alpha\ \wedge\ P\ \alpha\ \wedge\ (\forall \beta < \alpha.\ Ord\ \beta \longrightarrow \neg P\ \beta))$  (**is** *?lhs*  $\longleftrightarrow$  *?rhs*)

*<proof>*

**lemma** *Ord-mono-imp-increasing*:

**assumes**  $fun\text{-}hD: h \in D \rightarrow D$

**and**  $mono\text{-}h: strict\text{-}mono\text{-}on\ D\ h$

**and**  $D \subseteq ON$  **and**  $\nu: \nu \in D$

**shows**  $\nu \leq h\ \nu$

*<proof>*

**lemma** *le-Sup-iff*:

**assumes**  $A \subseteq ON$   $Ord\ x$  *small*  $A$  **shows**  $x \leq \bigsqcup A \longleftrightarrow (\forall y \in ON.\ y < x \longrightarrow (\exists a \in A.\ y < a))$

*<proof>*

**lemma** *le-SUP-iff*:  $\llbracket f\ ' A \subseteq ON; Ord\ x; small\ A \rrbracket \implies x \leq \bigsqcup (f\ ' A) \longleftrightarrow (\forall y \in ON.\ y < x \longrightarrow (\exists i \in A.\ y < f\ i))$

*<proof>*

## 1.5 Transfinite Recursion and the V-levels

**definition** *transrec* ::  $((V \Rightarrow 'a) \Rightarrow V \Rightarrow 'a) \Rightarrow V \Rightarrow 'a$   
**where** *transrec*  $H a \equiv wfrec \{(x,y). x \in elts y\} H a$

**lemma** *transrec*: *transrec*  $H a = H (\lambda x \in elts a. transrec H x) a$   
*<proof>*

Avoids explosions in proofs; resolve it with a meta-level definition

**lemma** *def-transrec*:

$\llbracket \bigwedge x. f x \equiv transrec H x \rrbracket \implies f a = H(\lambda x \in elts a. f x) a$   
*<proof>*

**lemma** *eps-induct* [*case-names step*]:

**assumes**  $\bigwedge x. (\bigwedge y. y \in elts x \implies P y) \implies P x$   
**shows**  $P a$   
*<proof>*

**definition** *Vfrom* ::  $[V, V] \Rightarrow V$

**where** *Vfrom*  $a \equiv transrec (\lambda f x. a \sqcup \bigsqcup ((\lambda y. VPow(f y)) ' elts x))$

**abbreviation** *Vset* ::  $V \Rightarrow V$  **where** *Vset*  $\equiv Vfrom 0$

**lemma** *Vfrom*: *Vfrom*  $a i = a \sqcup \bigsqcup ((\lambda j. VPow(Vfrom a j)) ' elts i)$   
*<proof>*

**lemma** *Vfrom-0* [*simp*]: *Vfrom*  $a 0 = a$   
*<proof>*

**lemma** *Vset*: *Vset*  $i = \bigsqcup ((\lambda j. VPow(Vset j)) ' elts i)$   
*<proof>*

**lemma** *Vfrom-mono1*:

**assumes**  $a \leq b$  **shows** *Vfrom*  $a i \leq Vfrom b i$   
*<proof>*

**lemma** *Vfrom-mono2*: *Vfrom*  $a i \leq Vfrom a (i \sqcup j)$   
*<proof>*

**lemma** *Vfrom-mono*:  $\llbracket Ord i; a \leq b; i \leq j \rrbracket \implies Vfrom a i \leq Vfrom b j$   
*<proof>*

**lemma** *Transset-Vfrom*: *Transset*( $A$ )  $\implies Transset(Vfrom A i)$   
*<proof>*

**lemma** *Transset-Vset* [*simp*]: *Transset*(*Vset*  $i$ )

*<proof>*

**lemma** *Vfrom-sup*:  $Vfrom\ a\ (i \sqcup j) = Vfrom\ a\ i \sqcup Vfrom\ a\ j$   
*<proof>*

**lemma** *Vfrom-succ-Ord*:  
**assumes** *Ord i* **shows**  $Vfrom\ a\ (succ\ i) = a \sqcup VPow(Vfrom\ a\ i)$   
*<proof>*

**lemma** *Vset-succ*:  $Ord\ i \implies Vset(succ(i)) = VPow(Vset(i))$   
*<proof>*

**lemma** *Vfrom-Sup*:  
**assumes**  $X \neq \{\}$  *small X*  
**shows**  $Vfrom\ a\ (Sup\ X) = (\bigsqcup_{y \in X}. Vfrom\ a\ y)$   
*<proof>*

**lemma** *Limit-Vfrom-eq*:  
 $Limit(i) \implies Vfrom\ a\ i = (\bigsqcup_{y \in elts\ i}. Vfrom\ a\ y)$   
*<proof>*

end

## 2 Cartesian products, Disjoint Sums, Ranks, Cardinals

**theory** *ZFC-Cardinals*  
**imports** *ZFC-in-HOL*

**begin**

**declare**  $[[coercion-enabled]]$   
**declare**  $[[coercion\ ord-of-nat :: nat \Rightarrow V]]$

### 2.1 Ordered Pairs

**lemma** *singleton-eq-iff*  $[iff]: set\ \{a\} = set\ \{b\} \longleftrightarrow a=b$   
*<proof>*

**lemma** *doubleton-eq-iff*:  $set\ \{a,b\} = set\ \{c,d\} \longleftrightarrow (a=c \wedge b=d) \vee (a=d \wedge b=c)$   
*<proof>*

**definition** *vpair*  $:: V \Rightarrow V \Rightarrow V$   
**where**  $vpair\ a\ b = set\ \{set\ \{a\}, set\ \{a,b\}\}$

**definition** *vfst*  $:: V \Rightarrow V$   
**where**  $vfst\ p \equiv THE\ x. \exists y. p = vpair\ x\ y$

**definition**  $vsnd :: V \Rightarrow V$   
**where**  $vsnd\ p \equiv THE\ y.\ \exists x.\ p = vpair\ x\ y$

**definition**  $vsplit :: [[V, V] \Rightarrow 'a, V] \Rightarrow 'a::\{\}$  — for pattern-matching  
**where**  $vsplit\ c \equiv \lambda p.\ c\ (vfst\ p)\ (vsnd\ p)$

**nonterminal**  $Vs$

**syntax** (ASCII)

-*Tuple*  $:: [V, Vs] \Rightarrow V$  ( $\langle\langle(-, / -)\rangle\rangle$ )  
-*hpattern*  $:: [pttrn, patterns] \Rightarrow pttrn$  ( $\langle\langle(-, / -)\rangle\rangle$ )

**syntax**

$:: V \Rightarrow Vs$  ( $\langle(-)\rangle$ )  
-*Enum*  $:: [V, Vs] \Rightarrow Vs$  ( $\langle(-, / -)\rangle$ )  
-*Tuple*  $:: [V, Vs] \Rightarrow V$  ( $\langle\langle(-, / -)\rangle\rangle$ )  
-*hpattern*  $:: [pttrn, patterns] \Rightarrow pttrn$  ( $\langle\langle(-, / -)\rangle\rangle$ )

**syntax-consts**

-*Enum* -*Tuple*  $\equiv vpair\ and$   
-*hpattern*  $\equiv vsplit$

**translations**

$\langle x, y, z \rangle \equiv \langle x, \langle y, z \rangle \rangle$   
 $\langle x, y \rangle \equiv CONST\ vpair\ x\ y$   
 $\langle x, y, z \rangle \equiv \langle x, \langle y, z \rangle \rangle$   
 $\lambda\langle x, y, zs \rangle.\ b \equiv CONST\ vsplit(\lambda x\ \langle y, zs \rangle.\ b)$   
 $\lambda\langle x, y \rangle.\ b \equiv CONST\ vsplit(\lambda x\ y.\ b)$

**lemma**  $vpair-def'$ :  $vpair\ a\ b = set\ \{set\ \{a, a\}, set\ \{a, b\}\}$   
 $\langle proof \rangle$

**lemma**  $vpair-iff$  [simp]:  $vpair\ a\ b = vpair\ a'\ b' \longleftrightarrow a = a' \wedge b = b'$   
 $\langle proof \rangle$

**lemmas**  $vpair-inject = vpair-iff$  [THEN  $iffD1$ , THEN  $conjE$ , elim!]

**lemma**  $vfst-conv$  [simp]:  $vfst\ \langle a, b \rangle = a$   
 $\langle proof \rangle$

**lemma**  $vsnd-conv$  [simp]:  $vsnd\ \langle a, b \rangle = b$   
 $\langle proof \rangle$

**lemma**  $vsplit$  [simp]:  $vsplit\ c\ \langle a, b \rangle = c\ a\ b$   
 $\langle proof \rangle$

**lemma**  $vpair-neq-fst$ :  $\langle a, b \rangle \neq a$   
 $\langle proof \rangle$

**lemma**  $vpair-neq-snd$ :  $\langle a, b \rangle \neq b$   
 $\langle proof \rangle$

**lemma** *vpair-nonzero* [simp]:  $\langle x, y \rangle \neq 0$   
*<proof>*

**lemma** *zero-notin-vpair*:  $0 \notin \text{elts } \langle x, y \rangle$   
*<proof>*

**lemma** *inj-on-vpair* [simp]: *inj-on*  $(\lambda(x, y). \langle x, y \rangle) A$   
*<proof>*

## 2.2 Generalized Cartesian product

**definition** *VSigma* ::  $V \Rightarrow (V \Rightarrow V) \Rightarrow V$   
**where**  $VSigma A B \equiv \text{set}(\bigcup x \in \text{elts } A. \bigcup y \in \text{elts } (B x). \{\langle x, y \rangle\})$

**abbreviation** *vtimes* **where**  $vtimes A B \equiv VSigma A (\lambda x. B)$

**definition** *pairs* ::  $V \Rightarrow (V * V)\text{set}$   
**where**  $pairs r \equiv \{(x, y). \langle x, y \rangle \in \text{elts } r\}$

**lemma** *pairs-iff-elts*:  $\langle x, y \rangle \in \text{pairs } z \iff \langle x, y \rangle \in \text{elts } z$   
*<proof>*

**lemma** *VSigma-iff* [simp]:  $\langle a, b \rangle \in \text{elts } (VSigma A B) \iff a \in \text{elts } A \wedge b \in \text{elts } (B a)$   
*<proof>*

**lemma** *VSigmaI* [intro!]:  $\llbracket a \in \text{elts } A; b \in \text{elts } (B a) \rrbracket \implies \langle a, b \rangle \in \text{elts } (VSigma A B)$   
*<proof>*

**lemmas** *VSigmaD1* = *VSigma-iff* [THEN *iffD1*, THEN *conjunct1*]

**lemmas** *VSigmaD2* = *VSigma-iff* [THEN *iffD1*, THEN *conjunct2*]

The general elimination rule

**lemma** *VSigmaE* [elim!]:  
**assumes**  $c \in \text{elts } (VSigma A B)$   
**obtains**  $x y$  **where**  $x \in \text{elts } A$   $y \in \text{elts } (B x)$   $c = \langle x, y \rangle$   
*<proof>*

**lemma** *VSigmaE2* [elim!]:  
**assumes**  $\langle a, b \rangle \in \text{elts } (VSigma A B)$  **obtains**  $a \in \text{elts } A$  **and**  $b \in \text{elts } (B a)$   
*<proof>*

**lemma** *VSigma-empty1* [simp]:  $VSigma 0 B = 0$   
*<proof>*

**lemma** *times-iff* [simp]:  $\langle a, b \rangle \in \text{elts } (vtimes A B) \iff a \in \text{elts } A \wedge b \in \text{elts } B$   
*<proof>*

**lemma** *timesI* [*intro!*]:  $\llbracket a \in \text{elts } A; b \in \text{elts } B \rrbracket \implies \langle a, b \rangle \in \text{elts } (\text{vtimes } A B)$   
 ⟨*proof*⟩

**lemma** *times-empty2* [*simp*]:  $\text{vtimes } A 0 = 0$   
 ⟨*proof*⟩

**lemma** *times-empty-iff*:  $\text{VSigma } A B = 0 \iff A=0 \vee (\forall x \in \text{elts } A. B x = 0)$   
 ⟨*proof*⟩

**lemma** *elts-VSigma*:  $\text{elts } (\text{VSigma } A B) = (\lambda(x,y). \text{vpair } x y) \text{ `Sigma } (\text{elts } A)$   
 $(\lambda x. \text{elts } (B x))$   
 ⟨*proof*⟩

**lemma** *small-Sigma* [*simp*]:  
**assumes** *A*: *small A* **and** *B*:  $\bigwedge x. x \in A \implies \text{small } (B x)$   
**shows** *small* (*Sigma A B*)  
 ⟨*proof*⟩

**lemma** *small-Times* [*simp*]:  
**assumes** *small A* *small B* **shows** *small* ( $A \times B$ )  
 ⟨*proof*⟩

**lemma** *small-Times-iff*:  $\text{small } (A \times B) \iff \text{small } A \wedge \text{small } B \vee A=\{\} \vee B=\{\}$   
 (**is** - = ?*rhs*)  
 ⟨*proof*⟩

## 2.3 Disjoint Sum

**definition** *vsum* ::  $V \Rightarrow V \Rightarrow V$  (**infixl**  $\langle \uplus \rangle$  65) **where**  
 $A \uplus B \equiv (\text{VSigma } (\text{set } \{0\}) (\lambda x. A)) \sqcup (\text{VSigma } (\text{set } \{1\}) (\lambda x. B))$

**definition** *Inl* ::  $V \Rightarrow V$  **where**  
 $\text{Inl } a \equiv \langle 0, a \rangle$

**definition** *Inr* ::  $V \Rightarrow V$  **where**  
 $\text{Inr } b \equiv \langle 1, b \rangle$

**lemmas** *sum-defs* = *vsum-def Inl-def Inr-def*

**lemma** *Inl-nonzero* [*simp*]:  $\text{Inl } x \neq 0$   
 ⟨*proof*⟩

**lemma** *Inr-nonzero* [*simp*]:  $\text{Inr } x \neq 0$   
 ⟨*proof*⟩

### 2.3.1 Equivalences for the injections and an elimination rule

**lemma** *Inl-in-sum-iff* [*iff*]:  $\text{Inl } a \in \text{elts } (A \uplus B) \iff a \in \text{elts } A$   
 ⟨*proof*⟩

**lemma** *Inr-in-sum-iff* [*iff*]:  $Inr\ b \in elts\ (A \uplus B) \longleftrightarrow b \in elts\ B$   
 ⟨*proof*⟩

**lemma** *sumE* [*elim!*]:  
**assumes**  $u: u \in elts\ (A \uplus B)$   
**obtains**  $x$  **where**  $x \in elts\ A\ u=Inl\ x \mid y$  **where**  $y \in elts\ B\ u=Inr\ y$  ⟨*proof*⟩

### 2.3.2 Injection and freeness equivalences, for rewriting

**lemma** *Inl-iff* [*iff*]:  $Inl\ a=Inl\ b \longleftrightarrow a=b$   
 ⟨*proof*⟩

**lemma** *Inr-iff* [*iff*]:  $Inr\ a=Inr\ b \longleftrightarrow a=b$   
 ⟨*proof*⟩

**lemma** *inj-on-Inl* [*simp*]: *inj-on*  $Inl\ A$   
 ⟨*proof*⟩

**lemma** *inj-on-Inr* [*simp*]: *inj-on*  $Inr\ A$   
 ⟨*proof*⟩

**lemma** *Inl-Inr-iff* [*iff*]:  $Inl\ a=Inr\ b \longleftrightarrow False$   
 ⟨*proof*⟩

**lemma** *Inr-Inl-iff* [*iff*]:  $Inr\ b=Inl\ a \longleftrightarrow False$   
 ⟨*proof*⟩

**lemma** *sum-empty* [*simp*]:  $0 \uplus 0 = 0$   
 ⟨*proof*⟩

**lemma** *elts-vsum*:  $elts\ (a \uplus b) = Inl\ ' (elts\ a) \cup Inr\ ' (elts\ b)$   
 ⟨*proof*⟩

**lemma** *sum-iff*:  $u \in elts\ (A \uplus B) \longleftrightarrow (\exists x. x \in elts\ A \wedge u=Inl\ x) \vee (\exists y. y \in elts\ B \wedge u=Inr\ y)$   
 ⟨*proof*⟩

**lemma** *sum-subset-iff*:  $A \uplus B \leq C \uplus D \longleftrightarrow A \leq C \wedge B \leq D$   
 ⟨*proof*⟩

**lemma** *sum-equal-iff*:  
**fixes**  $A :: V$  **shows**  $A \uplus B = C \uplus D \longleftrightarrow A=C \wedge B=D$   
 ⟨*proof*⟩

**definition** *is-sum* ::  $V \Rightarrow bool$   
**where**  $is-sum\ z = (\exists x. z = Inl\ x \vee z = Inr\ x)$

**definition** *sum-case* ::  $(V \Rightarrow 'a) \Rightarrow (V \Rightarrow 'a) \Rightarrow V \Rightarrow 'a$   
**where**

*sum-case f g a*  $\equiv$   
 THE  $z. (\forall x. a = \text{Inl } x \longrightarrow z = f x) \wedge (\forall y. a = \text{Inr } y \longrightarrow z = g y) \wedge (\neg \text{is-sum } a \longrightarrow z = \text{undefined})$

**lemma** *sum-case-Inl* [simp]: *sum-case f g (Inl x) = f x*  
 ⟨proof⟩

**lemma** *sum-case-Inr* [simp]: *sum-case f g (Inr y) = g y*  
 ⟨proof⟩

**lemma** *sum-case-non* [simp]:  $\neg \text{is-sum } a \implies \text{sum-case } f g a = \text{undefined}$   
 ⟨proof⟩

**lemma** *is-sum-cases*:  $(\exists x. z = \text{Inl } x \vee z = \text{Inr } x) \vee \neg \text{is-sum } z$   
 ⟨proof⟩

**lemma** *sum-case-split*:  
 $P (\text{sum-case } f g a) \longleftrightarrow (\forall x. a = \text{Inl } x \longrightarrow P(f x)) \wedge (\forall y. a = \text{Inr } y \longrightarrow P(g y)) \wedge (\neg \text{is-sum } a \longrightarrow P \text{ undefined})$   
 ⟨proof⟩

**lemma** *sum-case-split-asm*:  
 $P (\text{sum-case } f g a) \longleftrightarrow \neg ((\exists x. a = \text{Inl } x \wedge \neg P(f x)) \vee (\exists y. a = \text{Inr } y \wedge \neg P(g y)) \vee (\neg \text{is-sum } a \wedge \neg P \text{ undefined}))$   
 ⟨proof⟩

### 2.3.3 Applications of disjoint sums and pairs: general union theorems for small sets

**lemma** *small-Un*:  
 assumes  $X$ : *small X* and  $Y$ : *small Y*  
 shows *small (X  $\cup$  Y)*  
 ⟨proof⟩

**lemma** *small-UN* [simp,intro]:  
 assumes  $A$ : *small A* and  $B$ :  $\bigwedge x. x \in A \implies \text{small } (B x)$   
 shows *small ( $\bigcup_{x \in A}. B x$ )*  
 ⟨proof⟩

**lemma** *small-Union* [simp,intro]:  
 assumes  $\mathcal{A} \subseteq \text{Collect small small } A$   
 shows *small ( $\bigcup \mathcal{A}$ )*  
 ⟨proof⟩

## 2.4 Generalised function space and lambda

**definition** *VLambda* ::  $V \Rightarrow (V \Rightarrow V) \Rightarrow V$   
 where *VLambda A b*  $\equiv \text{set } ((\lambda x. \langle x, b \ x \rangle) \text{ `elts } A)$

**definition** *app* ::  $[V, V] \Rightarrow V$

**where**  $app\ f\ x \equiv THE\ y.\ \langle x, y \rangle \in elts\ f$

**lemma** *beta* [*simp*]:

**assumes**  $x \in elts\ A$

**shows**  $app\ (VLambda\ A\ b)\ x = b\ x$

*<proof>*

**definition**  $VPi :: V \Rightarrow (V \Rightarrow V) \Rightarrow V$

**where**  $VPi\ A\ B \equiv set\ \{f \in elts\ (VPow(VSigma\ A\ B)).\ elts\ A \leq Domain\ (pairs\ f) \wedge\ single\text{-valued}\ (pairs\ f)\}$

**lemma** *VPi-I*:

**assumes**  $\bigwedge x.\ x \in elts\ A \implies b\ x \in elts\ (B\ x)$

**shows**  $VLambda\ A\ b \in elts\ (VPi\ A\ B)$

*<proof>*

**lemma** *apply-pair*:

**assumes**  $f: f \in elts\ (VPi\ A\ B)$  **and**  $x: x \in elts\ A$

**shows**  $\langle x, app\ f\ x \rangle \in elts\ f$

*<proof>*

**lemma** *VPi-D*:

**assumes**  $f: f \in elts\ (VPi\ A\ B)$  **and**  $x: x \in elts\ A$

**shows**  $app\ f\ x \in elts\ (B\ x)$

*<proof>*

**lemma** *VPi-memberD*:

**assumes**  $f: f \in elts\ (VPi\ A\ B)$  **and**  $p: p \in elts\ f$

**obtains**  $x$  **where**  $x \in elts\ A$   $p = \langle x, app\ f\ x \rangle$

*<proof>*

**lemma** *fun-ext*:

**assumes**  $f \in elts\ (VPi\ A\ B)$   $g \in elts\ (VPi\ A\ B)$   $\bigwedge x.\ x \in elts\ A \implies app\ f\ x = app\ g\ x$

**shows**  $f = g$

*<proof>*

**lemma** *eta*[*simp*]:

**assumes**  $f \in elts\ (VPi\ A\ B)$

**shows**  $VLambda\ A\ ((app)\ f) = f$

*<proof>*

**lemma** *fst-pairs-VLambda*:  $fst\ 'pairs\ (VLambda\ A\ f) = elts\ A$

*<proof>*

**lemma** *snd-pairs-VLambda*:  $snd\ 'pairs\ (VLambda\ A\ f) = f\ 'elts\ A$

*<proof>*

**lemma** *VLambda-eq-D1*:  $VLambda A f = VLambda B g \implies A = B$   
 ⟨proof⟩

**lemma** *VLambda-eq-D2*:  $\llbracket VLambda A f = VLambda A g; x \in elts A \rrbracket \implies f x = g x$   
 ⟨proof⟩

## 2.5 Transitive closure of a set

**definition** *TC* ::  $V \Rightarrow V$   
 where  $TC \equiv transrec (\lambda f x. x \sqcup \sqcup (f \text{ ` } elts x))$

**lemma** *TC*:  $TC a = a \sqcup \sqcup (TC \text{ ` } elts a)$   
 ⟨proof⟩

**lemma** *TC-0* [*simp*]:  $TC 0 = 0$   
 ⟨proof⟩

**lemma** *arg-subset-TC*:  $a \leq TC a$   
 ⟨proof⟩

**lemma** *Transset-TC*:  $Transset(TC a)$   
 ⟨proof⟩

**lemma** *TC-least*:  $\llbracket Transset x; a \leq x \rrbracket \implies TC a \leq x$   
 ⟨proof⟩

**definition** *less-TC* (**infix**  $\langle \sqsubset \rangle$  50)  
 where  $x \sqsubset y \equiv x \in elts (TC y)$

**definition** *le-TC* (**infix**  $\langle \sqsubseteq \rangle$  50)  
 where  $x \sqsubseteq y \equiv x \sqsubset y \vee x=y$

**lemma** *less-TC-imp-not-le*:  $x \sqsubset a \implies \neg a \leq x$   
 ⟨proof⟩

**lemma** *non-TC-less-0* [*iff*]:  $\neg (x \sqsubset 0)$   
 ⟨proof⟩

**lemma** *less-TC-iff*:  $x \sqsubset y \longleftrightarrow (\exists z \in elts y. x \sqsubseteq z)$   
 ⟨proof⟩

**lemma** *nonzero-less-TC*:  $x \neq 0 \implies 0 \sqsubset x$   
 ⟨proof⟩

**lemma** *less-irrefl-TC* [*simp*]:  $\neg x \sqsubset x$   
 ⟨proof⟩

**lemma** *less-asym-TC*:  $\llbracket x \sqsubset y; y \sqsubset x \rrbracket \implies False$

*<proof>*

**lemma** *le-antisym-TC*:  $\llbracket x \sqsubseteq y; y \sqsubseteq x \rrbracket \implies x = y$   
*<proof>*

**lemma** *less-le-TC*:  $x \sqsubset y \iff x \sqsubseteq y \wedge x \neq y$   
*<proof>*

**lemma** *less-imp-le-TC* [*iff*]:  $x \sqsubset y \implies x \sqsubseteq y$   
*<proof>*

**lemma** *le-TC-refl* [*iff*]:  $x \sqsubseteq x$   
*<proof>*

**lemma** *le-TC-trans* [*trans*]:  $\llbracket x \sqsubseteq y; y \sqsubseteq z \rrbracket \implies x \sqsubseteq z$   
*<proof>*

**context** *order*  
**begin**

**lemma** *nless-le-TC*:  $(\neg a \sqsubset b) \iff (\neg a \sqsubseteq b) \vee a = b$   
*<proof>*

**lemma** *eq-refl-TC*:  $x = y \implies x \sqsubseteq y$   
*<proof>*

*<ML>*

**end**

**lemma** *less-TC-trans* [*trans*]:  $\llbracket x \sqsubset y; y \sqsubset z \rrbracket \implies x \sqsubset z$   
**and** *less-le-TC-trans*:  $\llbracket x \sqsubset y; y \sqsubseteq z \rrbracket \implies x \sqsubset z$   
**and** *le-less-TC-trans* [*trans*]:  $\llbracket x \sqsubseteq y; y \sqsubset z \rrbracket \implies x \sqsubset z$   
*<proof>*

**lemma** *TC-sup-distrib*:  $TC (x \sqcup y) = TC x \sqcup TC y$   
*<proof>*

**lemma** *TC-Sup-distrib*:  
**assumes** *small X shows*  $TC (\bigsqcup X) = \bigsqcup (TC \text{ ` } X)$   
*<proof>*

**lemma** *TC'*:  $TC x = x \sqcup TC (\bigsqcup (\text{elts } x))$   
*<proof>*

**lemma** *TC-eq-0-iff* [*simp*]:  $TC x = 0 \iff x = 0$   
*<proof>*

A distinctive induction principle

**lemma** *TC-induct-down-lemma*:  
**assumes** *ab*:  $a \sqsubset b$  **and** *base*:  $b \leq d$   
**and step**:  $\bigwedge y z. \llbracket y \sqsubset b; y \in \text{elts } d; z \in \text{elts } y \rrbracket \implies z \in \text{elts } d$   
**shows**  $a \in \text{elts } d$   
 $\langle \text{proof} \rangle$

**lemma** *TC-induct-down* [*consumes 1, case-names base step small*]:  
**assumes**  $a \sqsubset b$   
**and**  $\bigwedge y. y \in \text{elts } b \implies P y$   
**and**  $\bigwedge y z. \llbracket y \sqsubset b; P y; z \in \text{elts } y \rrbracket \implies P z$   
**and** *small* (*Collect P*)  
**shows**  $P a$   
 $\langle \text{proof} \rangle$

## 2.6 Rank of a set

**definition** *rank* ::  $V \Rightarrow V$   
**where**  $\text{rank } a \equiv \text{transrec } (\lambda f x. \text{set } (\bigcup y \in \text{elts } x. \text{elts } (\text{succ}(f y)))) a$

**lemma** *rank*:  $\text{rank } a = \text{set}(\bigcup y \in \text{elts } a. \text{elts } (\text{succ}(\text{rank } y)))$   
 $\langle \text{proof} \rangle$

**lemma** *rank-Sup*:  $\text{rank } a = \bigsqcup ((\lambda y. \text{succ}(\text{rank } y)) ` \text{elts } a)$   
 $\langle \text{proof} \rangle$

**lemma** *Ord-rank* [*simp*]:  $\text{Ord}(\text{rank } a)$   
 $\langle \text{proof} \rangle$

**lemma** *rank-of-Ord*:  $\text{Ord } i \implies \text{rank } i = i$   
 $\langle \text{proof} \rangle$

**lemma** *Ord-iff-rank*:  $\text{Ord } x \longleftrightarrow \text{rank } x = x$   
 $\langle \text{proof} \rangle$

**lemma** *rank-lt*:  $a \in \text{elts } b \implies \text{rank } a < \text{rank } b$   
 $\langle \text{proof} \rangle$

**lemma** *rank-0* [*simp*]:  $\text{rank } 0 = 0$   
 $\langle \text{proof} \rangle$

**lemma** *rank-succ* [*simp*]:  $\text{rank}(\text{succ } x) = \text{succ}(\text{rank } x)$   
 $\langle \text{proof} \rangle$

**lemma** *rank-mono*:  $a \leq b \implies \text{rank } a \leq \text{rank } b$   
 $\langle \text{proof} \rangle$

**lemma** *VsetI*:  $\text{rank } b \sqsubset i \implies b \in \text{elts } (\text{Vset } i)$   
 $\langle \text{proof} \rangle$

**lemma** *Ord-VsetI*:  $\llbracket \text{Ord } i; \text{rank } b < i \rrbracket \implies b \in \text{elts } (\text{Vset } i)$   
 ⟨proof⟩

**lemma** *arg-le-Vset-rank*:  $a \leq \text{Vset}(\text{rank } a)$   
 ⟨proof⟩

**lemma** *two-in-Vset*:  
**obtains**  $\alpha$  **where**  $x \in \text{elts } (\text{Vset } \alpha)$   $y \in \text{elts } (\text{Vset } \alpha)$   
 ⟨proof⟩

**lemma** *rank-eq-0-iff* [*simp*]:  $\text{rank } x = 0 \longleftrightarrow x=0$   
 ⟨proof⟩

**lemma** *small-ranks-imp-small*:  
**assumes** *small* ( $\text{rank } 'A$ ) **shows** *small*  $A$   
 ⟨proof⟩

**lemma** *rank-Union*:  $\text{rank}(\bigsqcup A) = \bigsqcup (\text{rank } 'A)$   
 ⟨proof⟩

**lemma** *small-bounded-rank*: *small*  $\{x. \text{rank } x \in \text{elts } a\}$   
 ⟨proof⟩

**lemma** *small-bounded-rank-le*: *small*  $\{x. \text{rank } x \leq a\}$   
 ⟨proof⟩

**lemma** *TC-rank-lt*:  $a \sqsubset b \implies \text{rank } a < \text{rank } b$   
 ⟨proof⟩

**lemma** *TC-rank-mem*:  $x \sqsubset y \implies \text{rank } x \in \text{elts } (\text{rank } y)$   
 ⟨proof⟩

**lemma** *wf-TC-less*: *wf*  $\{(x,y). x \sqsubset y\}$   
 ⟨proof⟩

**lemma** *less-TC-minimal*:  
**assumes**  $P a$   
**obtains**  $x$  **where**  $P x$   $x \sqsubseteq a \wedge y. y \sqsubset x \implies \neg P y$   
 ⟨proof⟩

**lemma** *Vfrom-rank-eq*:  $\text{Vfrom } A (\text{rank}(x)) = \text{Vfrom } A x$   
 ⟨proof⟩

**lemma** *Vfrom-succ*:  $\text{Vfrom } A (\text{succ}(i)) = A \sqcup \text{VPow}(\text{Vfrom } A i)$   
 ⟨proof⟩

**lemma** *Vset-succ-TC*:  
**assumes**  $x \in \text{elts } (\text{Vset } (\text{ZFC-in-HOL.succ } k))$   $u \sqsubset x$   
**shows**  $u \in \text{elts } (\text{Vset } k)$

*<proof>*

## 2.7 Cardinal Numbers

We extend the membership relation to a wellordering

**definition**  $VWO :: (V \times V)$  set

**where**  $VWO \equiv @r. \{(x,y). x \in elts\ y\} \subseteq r \wedge Well\text{-}order\ r \wedge Field\ r = UNIV$

**lemma**  $VWO: \{(x,y). x \in elts\ y\} \subseteq VWO \wedge Well\text{-}order\ VWO \wedge Field\ VWO = UNIV$

*<proof>*

**lemma**  $wf\text{-}VWO: wf(VWO - Id)$

*<proof>*

**lemma**  $wf\text{-}Ord\text{-}less: wf\ \{(x, y). Ord\ y \wedge x < y\}$

*<proof>*

**lemma**  $refl\text{-}VWO: refl\ VWO$

*<proof>*

**lemma**  $trans\text{-}VWO: trans\ VWO$

*<proof>*

**lemma**  $antisym\text{-}VWO: antisym\ VWO$

*<proof>*

**lemma**  $total\text{-}VWO: total\ VWO$

*<proof>*

**lemma**  $total\text{-}VWOId: total\ (VWO - Id)$

*<proof>*

**lemma**  $Linear\text{-}order\text{-}VWO: Linear\text{-}order\ VWO$

*<proof>*

**lemma**  $wo\text{-}rel\text{-}VWO: wo\text{-}rel\ VWO$

*<proof>*

### 2.7.1 Transitive Closure and VWO

**lemma**  $mem\text{-}imp\text{-}VWO: x \in elts\ y \implies (x,y) \in VWO$

*<proof>*

**lemma**  $less\text{-}TC\text{-}imp\text{-}VWO: x \sqsubset y \implies (x,y) \in VWO$

*<proof>*

**lemma**  $le\text{-}TC\text{-}imp\text{-}VWO: x \sqsubseteq y \implies (x,y) \in VWO$

*<proof>*

**lemma** *le-TC-0-iff* [*simp*]:  $x \sqsubseteq 0 \longleftrightarrow x = 0$   
 ⟨*proof*⟩

**lemma** *less-TC-succ*:  $x \sqsubset \text{succ } \beta \longleftrightarrow x \sqsubset \beta \vee x = \beta$   
 ⟨*proof*⟩

**lemma** *le-TC-succ*:  $x \sqsubseteq \text{succ } \beta \longleftrightarrow x \sqsubseteq \beta \vee x = \text{succ } \beta$   
 ⟨*proof*⟩

**lemma** *Transset-TC-eq* [*simp*]:  $\text{Transset } x \Longrightarrow \text{TC } x = x$   
 ⟨*proof*⟩

**lemma** *Ord-TC-less-iff*:  $\llbracket \text{Ord } \alpha; \text{Ord } \beta \rrbracket \Longrightarrow \beta \sqsubset \alpha \longleftrightarrow \beta < \alpha$   
 ⟨*proof*⟩

**lemma** *Ord-mem-iff-less-TC*:  $\text{Ord } l \Longrightarrow k \in \text{elts } l \longleftrightarrow k \sqsubset l$   
 ⟨*proof*⟩

**lemma** *le-TC-Ord*:  $\llbracket \beta \sqsubseteq \alpha; \text{Ord } \alpha \rrbracket \Longrightarrow \text{Ord } \beta$   
 ⟨*proof*⟩

**lemma** *Ord-less-TC-mem*:  
**assumes**  $\text{Ord } \alpha$   $\beta \sqsubset \alpha$  **shows**  $\beta \in \text{elts } \alpha$   
 ⟨*proof*⟩

**lemma** *VWO-TC-le*:  $\llbracket \text{Ord } \alpha; \text{Ord } \beta; (\beta, \alpha) \in \text{VWO} \rrbracket \Longrightarrow \beta \sqsubseteq \alpha$   
 ⟨*proof*⟩

**lemma** *VWO-iff-Ord-le* [*simp*]:  $\llbracket \text{Ord } \alpha; \text{Ord } \beta \rrbracket \Longrightarrow (\beta, \alpha) \in \text{VWO} \longleftrightarrow \beta \leq \alpha$   
 ⟨*proof*⟩

**lemma** *zero-TC-le* [*iff*]:  $0 \sqsubseteq y$   
 ⟨*proof*⟩

**lemma** *succ-le-TC-iff*:  $\text{Ord } j \Longrightarrow \text{succ } i \sqsubseteq j \longleftrightarrow i \sqsubset j$   
 ⟨*proof*⟩

**lemma** *VWO-0-iff* [*simp*]:  $(x, 0) \in \text{VWO} \longleftrightarrow x = 0$   
 ⟨*proof*⟩

**lemma** *VWO-antisym*:  
**assumes**  $(x, y) \in \text{VWO}$   $(y, x) \in \text{VWO}$  **shows**  $x = y$   
 ⟨*proof*⟩

## 2.7.2 Relation VWF

**definition** *VWF* where  $\text{VWF} \equiv \text{VWO} - \text{Id}$

**lemma** *wf-VWF* [*iff*]: *wf VWF*  
⟨*proof*⟩

**lemma** *trans-VWF* [*iff*]: *trans VWF*  
⟨*proof*⟩

**lemma** *asym-VWF* [*iff*]: *asym VWF*  
⟨*proof*⟩

**lemma** *total-VWF* [*iff*]: *total VWF*  
⟨*proof*⟩

**lemma** *total-on-VWF* [*iff*]: *total-on A VWF*  
⟨*proof*⟩

**lemma** *VWF-asym*:  
assumes  $(x,y) \in VWF$   $(y,x) \in VWF$  shows *False*  
⟨*proof*⟩

**lemma** *VWF-non-refl* [*iff*]:  $(x,x) \notin VWF$   
⟨*proof*⟩

**lemma** *VWF-iff-Ord-less* [*simp*]:  $[[Ord\ \alpha; Ord\ \beta]] \implies (\alpha,\beta) \in VWF \longleftrightarrow \alpha < \beta$   
⟨*proof*⟩

**lemma** *mem-imp-VWF*:  $x \in elts\ y \implies (x,y) \in VWF$   
⟨*proof*⟩

## 2.8 Order types

**definition** *ordermap* ::  $'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow 'a \Rightarrow V$   
where *ordermap*  $A\ r \equiv wfrec\ r\ (\lambda f\ x.\ set\ (f\ ' \{y \in A.\ (y,x) \in r\}))$

**definition** *ordertype* ::  $'a\ set \Rightarrow ('a \times 'a)\ set \Rightarrow V$   
where *ordertype*  $A\ r \equiv set\ (ordermap\ A\ r\ ' A)$

**lemma** *ordermap-type*:  
 $small\ A \implies ordermap\ A\ r \in A \rightarrow elts\ (ordertype\ A\ r)$   
⟨*proof*⟩

**lemma** *ordermap-in-ordertype* [*intro*]:  $[[a \in A; small\ A]] \implies ordermap\ A\ r\ a \in elts\ (ordertype\ A\ r)$   
⟨*proof*⟩

**lemma** *ordermap*:  $wf\ r \implies ordermap\ A\ r\ a = set\ (ordermap\ A\ r\ ' \{y \in A.\ (y,a) \in r\})$   
⟨*proof*⟩

**lemma** *wf-Ord-ordermap* [*iff*]: assumes *wf r trans r* shows *Ord (ordermap A r)*

$x$ )  
 $\langle proof \rangle$

**lemma** *wf-Ord-ordertype*: **assumes**  $wf\ r\ trans\ r$  **shows**  $Ord(ordertype\ A\ r)$   
 $\langle proof \rangle$

**lemma** *Ord-ordertype [simp]*:  $Ord(ordertype\ A\ VWF)$   
 $\langle proof \rangle$

**lemma** *Ord-ordermap [simp]*:  $Ord\ (ordermap\ A\ VWF\ x)$   
 $\langle proof \rangle$

**lemma** *ordertype-singleton [simp]*:  
**assumes**  $wf\ r$   
**shows**  $ordertype\ \{x\}\ r = 1$   
 $\langle proof \rangle$

### 2.8.1 *ordermap* preserves the orderings in both directions

**lemma** *ordermap-mono*:  
**assumes**  $wx: (w, x) \in r$  **and**  $wf\ r\ w \in A\ small\ A$   
**shows**  $ordermap\ A\ r\ w \in elts\ (ordermap\ A\ r\ x)$   
 $\langle proof \rangle$

**lemma** *converse-ordermap-mono*:  
**assumes**  $ordermap\ A\ r\ y \in elts\ (ordermap\ A\ r\ x)\ wf\ r\ total-on\ A\ r\ x \in A\ y \in A\ small\ A$   
**shows**  $(y, x) \in r$   
 $\langle proof \rangle$

**lemma** *converse-ordermap-mono-iff*:  
**assumes**  $wf\ r\ total-on\ A\ r\ x \in A\ y \in A\ small\ A$   
**shows**  $ordermap\ A\ r\ y \in elts\ (ordermap\ A\ r\ x) \longleftrightarrow (y, x) \in r$   
 $\langle proof \rangle$

**lemma** *ordermap-surj*:  $elts\ (ordertype\ A\ r) \subseteq ordermap\ A\ r\ `A$   
 $\langle proof \rangle$

**lemma** *ordermap-bij*:  
**assumes**  $wf\ r\ total-on\ A\ r\ small\ A$   
**shows**  $bij\ betw\ (ordermap\ A\ r)\ A\ (elts\ (ordertype\ A\ r))$   
 $\langle proof \rangle$

**lemma** *ordermap-eq-iff [simp]*:  
 $\llbracket x \in A; y \in A; wf\ r; total-on\ A\ r; small\ A \rrbracket \implies ordermap\ A\ r\ x = ordermap\ A\ r\ y \longleftrightarrow x = y$   
 $\langle proof \rangle$

**lemma** *inv-into-ordermap*:  $\alpha \in elts\ (ordertype\ A\ r) \implies inv-into\ A\ (ordermap\ A$

$r$ )  $\alpha \in A$   
 $\langle proof \rangle$

**lemma** *ordertype-nat-imp-finite*:

**assumes** *ordertype A r = ord-of-nat m small A wf r total-on A r*  
**shows** *finite A*

$\langle proof \rangle$

**lemma** *wf-ordertype-epoll*:

**assumes** *wf r total-on A r small A*  
**shows** *elts (ordertype A r)  $\approx$  A*

$\langle proof \rangle$

**lemma** *ordertype-epoll*:

**assumes** *small A*  
**shows** *elts (ordertype A VWF)  $\approx$  A*

$\langle proof \rangle$

## 2.9 More advanced *ordertype* and *ordermap* results

**lemma** *ordermap-VWF-0 [simp]*: *ordermap A VWF 0 = 0*

$\langle proof \rangle$

**lemma** *ordertype-empty [simp]*: *ordertype {} r = 0*

$\langle proof \rangle$

**lemma** *ordertype-eq-0-iff [simp]*:  $\llbracket \text{small } X; \text{wf } r \rrbracket \implies \text{ordertype } X \text{ } r = 0 \longleftrightarrow X = \{\}$

$\langle proof \rangle$

**lemma** *ordermap-mono-less*:

**assumes**  $(w, x) \in r$   
**and** *wf r trans r*  
**and**  $w \in A \ x \in A$   
**and** *small A*  
**shows** *ordermap A r w < ordermap A r x*

$\langle proof \rangle$

**lemma** *ordermap-mono-le*:

**assumes**  $(w, x) \in r \vee w=x$   
**and** *wf r trans r*  
**and**  $w \in A \ x \in A$   
**and** *small A*  
**shows** *ordermap A r w  $\leq$  ordermap A r x*

$\langle proof \rangle$

**lemma** *converse-ordermap-le-mono*:

**assumes** *ordermap A r y  $\leq$  ordermap A r x wf r total r x  $\in$  A small A*  
**shows**  $(y, x) \in r \vee y=x$

*<proof>*

**lemma** *ordertype-mono:*

**assumes**  $X \subseteq Y$  **and**  $r: wf\ r\ trans\ r$  **and** *small*  $Y$

**shows**  $ordertype\ X\ r \leq ordertype\ Y\ r$

*<proof>*

**corollary** *ordertype-VWF-mono:*

**assumes**  $X \subseteq Y$  *small*  $Y$

**shows**  $ordertype\ X\ VWF \leq ordertype\ Y\ VWF$

*<proof>*

**lemma** *ordertype-UNION-ge:*

**assumes**  $A \in \mathcal{A}$   $wf\ r\ trans\ r$   $\mathcal{A} \subseteq Collect\ small\ small\ \mathcal{A}$

**shows**  $ordertype\ A\ r \leq ordertype\ (\bigcup \mathcal{A})\ r$

*<proof>*

**lemma** *inv-ordermap-mono-less:*

**assumes**  $(inv\ into\ M\ (ordermap\ M\ r)\ \alpha, inv\ into\ M\ (ordermap\ M\ r)\ \beta) \in r$

**and** *small*  $M$  **and**  $\alpha: \alpha \in elts\ (ordertype\ M\ r)$  **and**  $\beta: \beta \in elts\ (ordertype\ M\ r)$

**and**  $wf\ r\ trans\ r$

**shows**  $\alpha < \beta$

*<proof>*

**lemma** *inv-ordermap-mono-eq:*

**assumes**  $inv\ into\ M\ (ordermap\ M\ r)\ \alpha = inv\ into\ M\ (ordermap\ M\ r)\ \beta$

**and**  $\alpha \in elts\ (ordertype\ M\ r)$   $\beta \in elts\ (ordertype\ M\ r)$

**shows**  $\alpha = \beta$

*<proof>*

**lemma** *inv-ordermap-VWF-mono-le:*

**assumes**  $inv\ into\ M\ (ordermap\ M\ VWF)\ \alpha \leq inv\ into\ M\ (ordermap\ M\ VWF)\ \beta$

**and**  $M \subseteq ON$  *small*  $M$  **and**  $\alpha: \alpha \in elts\ (ordertype\ M\ VWF)$  **and**  $\beta: \beta \in elts\ (ordertype\ M\ VWF)$

**shows**  $\alpha \leq \beta$

*<proof>*

**lemma** *inv-ordermap-VWF-mono-iff:*

**assumes**  $M \subseteq ON$  *small*  $M$  **and**  $\alpha \in elts\ (ordertype\ M\ VWF)$  **and**  $\beta \in elts\ (ordertype\ M\ VWF)$

**shows**  $inv\ into\ M\ (ordermap\ M\ VWF)\ \alpha \leq inv\ into\ M\ (ordermap\ M\ VWF)\ \beta$   
 $\iff \alpha \leq \beta$

*<proof>*

**lemma** *inv-ordermap-VWF-strict-mono-iff:*

**assumes**  $M \subseteq ON$  *small*  $M$  **and**  $\alpha \in elts\ (ordertype\ M\ VWF)$  **and**  $\beta \in elts\ (ordertype\ M\ VWF)$

**shows**  $inv\ into\ M\ (ordermap\ M\ VWF)\ \alpha < inv\ into\ M\ (ordermap\ M\ VWF)\ \beta$

$\longleftrightarrow \alpha < \beta$   
 ⟨proof⟩

**lemma** *strict-mono-on-ordertype*:

**assumes**  $M \subseteq ON$  *small*  $M$

**obtains**  $f$  **where**  $f \in \text{elts}(\text{ordertype } M \text{ VWF}) \rightarrow M$  *strict-mono-on* ( $\text{elts}(\text{ordertype } M \text{ VWF})$ )  $f$

⟨proof⟩

**lemma** *ordermap-inc-eg*:

**assumes**  $x \in A$  *small*  $A$

**and**  $\pi: \bigwedge x y. \llbracket x \in A; y \in A; (x,y) \in r \rrbracket \implies (\pi x, \pi y) \in s$

**and**  $r: \text{wf } r$  *total-on*  $A$   $r$  **and**  $\text{wf } s$

**shows**  $\text{ordermap}(\pi \text{ ' } A) s(\pi x) = \text{ordermap } A r x$

⟨proof⟩

**lemma** *ordertype-inc-eg*:

**assumes** *small*  $A$

**and**  $\pi: \bigwedge x y. \llbracket x \in A; y \in A; (x,y) \in r \rrbracket \implies (\pi x, \pi y) \in s$

**and**  $r: \text{wf } r$  *total-on*  $A$   $r$  **and**  $\text{wf } s$

**shows**  $\text{ordertype}(\pi \text{ ' } A) s = \text{ordertype } A r$

⟨proof⟩

**lemma** *ordertype-inc-le*:

**assumes** *small*  $A$  *small*  $B$

**and**  $\pi: \bigwedge x y. \llbracket x \in A; y \in A; (x,y) \in r \rrbracket \implies (\pi x, \pi y) \in s$

**and**  $r: \text{wf } r$  *total-on*  $A$   $r$  **and**  $\text{wf } s$  *trans*  $s$

**and**  $\pi \text{ ' } A \subseteq B$

**shows**  $\text{ordertype } A r \leq \text{ordertype } B s$

⟨proof⟩

**corollary** *ordertype-VWF-inc-eg*:

**assumes**  $A \subseteq ON$   $\pi \text{ ' } A \subseteq ON$  *small*  $A$  **and**  $\bigwedge x y. \llbracket x \in A; y \in A; x < y \rrbracket \implies \pi x < \pi y$

**shows**  $\text{ordertype}(\pi \text{ ' } A) \text{ VWF} = \text{ordertype } A \text{ VWF}$

⟨proof⟩

**lemma** *ordertype-image-ordermap*:

**assumes** *small*  $A$   $X \subseteq A$   $\text{wf } r$  *trans*  $r$  *total-on*  $X$   $r$

**shows**  $\text{ordertype}(\text{ordermap } A r \text{ ' } X) \text{ VWF} = \text{ordertype } X r$

⟨proof⟩

**lemma** *ordertype-map-image*:

**assumes**  $B \subseteq A$  *small*  $A$

**shows**  $\text{ordertype}(\text{ordermap } A \text{ VWF ' } A - \text{ordermap } A \text{ VWF ' } B) \text{ VWF} = \text{ordertype}(A - B) \text{ VWF}$

⟨proof⟩

**proposition** *ordertype-le-ordertype*:

**assumes**  $r$ :  $wf\ r\ total\ on\ A\ r$  **and**  $small\ A$   
**assumes**  $s$ :  $wf\ s\ total\ on\ B\ s\ trans\ s$  **and**  $small\ B$   
**shows**  $ordertype\ A\ r \leq ordertype\ B\ s \longleftrightarrow$   
 $(\exists f \in A \rightarrow B. inj\ on\ f\ A \wedge (\forall x \in A. \forall y \in A. ((x,y) \in r \longrightarrow (f\ x, f\ y) \in s)))$   
**(is ?lhs = ?rhs)**  
 $\langle proof \rangle$

**lemma**  $iso\ imp\ ordertype\ eq\ ordertype$ :  
**assumes**  $iso$ :  $iso\ r\ r'\ f$   
**and**  $wf\ r$   
**and**  $Total\ r$   
**and**  $sm$ :  $small\ (Field\ r)$   
**shows**  $ordertype\ (Field\ r)\ r = ordertype\ (Field\ r')\ r'$   
 $\langle proof \rangle$

**lemma**  $ordertype\ infinite\ ge\ \omega$ :  
**assumes**  $infinite\ A\ small\ A$   
**shows**  $ordertype\ A\ VWF \geq \omega$   
 $\langle proof \rangle$

**lemma**  $ordertype\ eqI$ :  
**assumes**  $wf\ r\ total\ on\ A\ r\ small\ A\ wf\ s$   
 $bij\ betw\ f\ A\ B\ (\forall x \in A. \forall y \in A. (f\ x, f\ y) \in s \longleftrightarrow (x,y) \in r)$   
**shows**  $ordertype\ A\ r = ordertype\ B\ s$   
 $\langle proof \rangle$

**lemma**  $ordermap\ eq\ self$ :  
**assumes**  $Ord\ \alpha$  **and**  $x$ :  $x \in elts\ \alpha$   
**shows**  $ordermap\ (elts\ \alpha)\ VWF\ x = x$   
 $\langle proof \rangle$

**lemma**  $ordertype\ eq\ Ord\ [simp]$ :  
**assumes**  $Ord\ \alpha$   
**shows**  $ordertype\ (elts\ \alpha)\ VWF = \alpha$   
 $\langle proof \rangle$

**proposition**  $ordertype\ eq\ iff$ :  
**assumes**  $\alpha$ :  $Ord\ \alpha$  **and**  $r$ :  $wf\ r$  **and**  $small\ A\ total\ on\ A\ r\ trans\ r$   
**shows**  $ordertype\ A\ r = \alpha \longleftrightarrow$   
 $(\exists f. bij\ betw\ f\ A\ (elts\ \alpha) \wedge (\forall x \in A. \forall y \in A. f\ x < f\ y \longleftrightarrow (x,y) \in r))$   
**(is ?lhs = ?rhs)**  
 $\langle proof \rangle$

**corollary**  $ordertype\ VWF\ eq\ iff$ :  
**assumes**  $Ord\ \alpha\ small\ A$   
**shows**  $ordertype\ A\ VWF = \alpha \longleftrightarrow$   
 $(\exists f. bij\ betw\ f\ A\ (elts\ \alpha) \wedge (\forall x \in A. \forall y \in A. f\ x < f\ y \longleftrightarrow (x,y) \in VWF))$

*<proof>*

**lemma** *ordertype-le-Ord*:  
assumes  $Ord\ \alpha\ X \subseteq elts\ \alpha$   
shows  $ordertype\ X\ VWF \leq \alpha$   
*<proof>*

**lemma** *ordertype-inc-le-Ord*:  
assumes  $small\ A\ Ord\ \alpha$   
and  $\pi: \bigwedge x\ y. \llbracket x \in A; y \in A; (x,y) \in r \rrbracket \implies \pi\ x < \pi\ y$   
and  $wf\ r\ total-on\ A\ r$   
and  $sub: \pi\ ` A \subseteq elts\ \alpha$   
shows  $ordertype\ A\ r \leq \alpha$   
*<proof>*

**lemma** *le-ordertype-obtains-subset*:  
assumes  $\alpha: \beta \leq \alpha\ ordertype\ H\ VWF = \alpha$  and  $small\ H\ Ord\ \beta$   
obtains  $G$  where  $G \subseteq H\ ordertype\ G\ VWF = \beta$   
*<proof>*

**lemma** *ordertype-infinite- $\omega$* :  
assumes  $A \subseteq elts\ \omega\ infinite\ A$   
shows  $ordertype\ A\ VWF = \omega$   
*<proof>*

For infinite sets of natural numbers

**lemma** *ordertype-nat- $\omega$* :  
assumes  $infinite\ N$  shows  $ordertype\ N\ less-than = \omega$   
*<proof>*

**proposition** *ordertype-eq-ordertype*:  
assumes  $r: wf\ r\ total-on\ A\ r\ trans\ r$  and  $small\ A$   
assumes  $s: wf\ s\ total-on\ B\ s\ trans\ s$  and  $small\ B$   
shows  $ordertype\ A\ r = ordertype\ B\ s \longleftrightarrow$   
 $(\exists f. bij-betw\ f\ A\ B \wedge (\forall x \in A. \forall y \in A. (f\ x, f\ y) \in s \longleftrightarrow (x,y) \in r))$   
(is ?lhs = ?rhs)  
*<proof>*

**corollary** *ordertype-eq-ordertype-iso*:  
assumes  $r: wf\ r\ total-on\ A\ r\ trans\ r$  and  $small\ A$  and  $FA: Field\ r = A$   
assumes  $s: wf\ s\ total-on\ B\ s\ trans\ s$  and  $small\ B$  and  $FB: Field\ s = B$   
shows  $ordertype\ A\ r = ordertype\ B\ s \longleftrightarrow (\exists f. iso\ r\ s\ f)$   
(is ?lhs = ?rhs)  
*<proof>*

**lemma** *Limit-ordertype-imp-Field-Restr*:  
assumes  $Lim: Limit\ (ordertype\ A\ r)$  and  $r: wf\ r\ total-on\ A\ r$  and  $small\ A$   
shows  $Field\ (Restr\ r\ A) = A$

*<proof>*

**lemma** *ordertype-Field-Restr:*

**assumes** *wf r total-on A r trans r small A Field (Restr r A) = A*  
**shows** *ordertype (Field (Restr r A)) (Restr r A) = ordertype A r*  
*<proof>*

**proposition** *ordertype-eq-ordertype-iso-Restr:*

**assumes** *r: wf r total-on A r trans r and small A and FA: Field (Restr r A) = A*  
**assumes** *s: wf s total-on B s trans s and small B and FB: Field (Restr s B) = B*  
**shows** *ordertype A r = ordertype B s  $\longleftrightarrow$  ( $\exists f. iso (Restr r A) (Restr s B) f$ )*  
*(is ?lhs = ?rhs)*  
*<proof>*

**lemma** *ordermap-insert:*

**assumes** *Ord  $\alpha$  and y: Ord y  $y \leq \alpha$  and U:  $U \subseteq elts \alpha$*   
**shows** *ordermap (insert  $\alpha$  U) VWF y = ordermap U VWF y*  
*<proof>*

**lemma** *ordertype-insert:*

**assumes** *Ord  $\alpha$  and U:  $U \subseteq elts \alpha$*   
**shows** *ordertype (insert  $\alpha$  U) VWF = succ (ordertype U VWF)*  
*<proof>*

**lemma** *finite-ordertype-le-card:*

**assumes** *finite A wf r trans r*  
**shows** *ordertype A r  $\leq$  ord-of-nat (card A)*  
*<proof>*

**lemma** *ordertype-VWF- $\omega$ :*

**assumes** *finite A*  
**shows** *ordertype A VWF  $\in$  elts  $\omega$*   
*<proof>*

**lemma** *ordertype-VWF-finite-nat:*

**assumes** *finite A*  
**shows** *ordertype A VWF = ord-of-nat (card A)*  
*<proof>*

**lemma** *finite-ordertype-eq-card:*

**assumes** *small A wf r trans r total-on A r*  
**shows** *ordertype A r = ord-of-nat m  $\longleftrightarrow$  finite A  $\wedge$  card A = m*  
*<proof>*

**lemma** *ex-bij-betw-strict-mono-card:*

**assumes** *finite M  $M \subseteq ON$*

**obtains**  $h$  **where**  $\text{bij-betw } h \{..<\text{card } M\} M$  **and**  $\text{strict-mono-on } \{..<\text{card } M\} h$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{ordertype-finite-less-than}$  [simp]:  
**assumes**  $\text{finite } A$  **shows**  $\text{ordertype } A \text{ less-than} = \text{card } A$   
 $\langle \text{proof} \rangle$

## 2.10 Cardinality of an arbitrary HOL set

**definition**  $\text{gcard} :: 'a \text{ set} \Rightarrow V$   
**where**  $\text{gcard } X \equiv \text{if small } X \text{ then } (\text{LEAST } i. \text{Ord } i \wedge \text{elts } i \approx X) \text{ else } 0$

## 2.11 Cardinality of a set

**definition**  $\text{vcard} :: V \Rightarrow V$   
**where**  $\text{vcard } a \equiv (\text{LEAST } i. \text{Ord } i \wedge \text{elts } i \approx \text{elts } a)$

**lemma**  $\text{gcard-eq-vcard}$  [simp]:  $\text{gcard } (\text{elts } x) = \text{vcard } x$   
 $\langle \text{proof} \rangle$

**definition**  $\text{Card} :: V \Rightarrow \text{bool}$   
**where**  $\text{Card } i \equiv i = \text{vcard } i$

**abbreviation**  $\text{CARD}$  **where**  $\text{CARD} \equiv \text{Collect } \text{Card}$

**lemma**  $\text{cardinal-cong}$ :  $\text{elts } x \approx \text{elts } y \implies \text{vcard } x = \text{vcard } y$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{gcardinal-cong}$ :  
**assumes**  $X \approx Y$  **shows**  $\text{gcard } X = \text{gcard } Y$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{vcard-set-image}$ :  $\text{inj-on } f \ (\text{elts } x) \implies \text{vcard } (\text{set } (f \ ' \ \text{elts } x)) = \text{vcard } x$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{gcard-image}$ :  $\text{inj-on } f \ X \implies \text{gcard } (f \ ' \ X) = \text{gcard } X$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{Card-cardinal-eq}$ :  $\text{Card } \kappa \implies \text{vcard } \kappa = \kappa$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{Card-is-Ord}$ :  
**assumes**  $\text{Card } \kappa$  **shows**  $\text{Ord } \kappa$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{cardinal-epoll}$ :  $\text{elts } (\text{vcard } a) \approx \text{elts } a$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{inj-into-vcard}$ :  
**obtains**  $f$  **where**  $f \in \text{elts } A \rightarrow \text{elts } (\text{vcard } A) \text{ inj-on } f \ (\text{elts } A)$

*<proof>*

**lemma** *cardinal-idem* [*simp*]:  $vcard (vcard a) = vcard a$   
*<proof>*

**lemma** *subset-smaller-vc*:  
**assumes**  $\kappa \leq vcard x$  *Card*  $\kappa$   
**obtains**  $y$  **where**  $y \leq x$   $vcard y = \kappa$   
*<proof>*

every natural number is a (finite) cardinal

**lemma** *nat-into-Card*:  
**assumes**  $\alpha \in elts \omega$  **shows** *Card*( $\alpha$ )  
*<proof>*

**lemma** *Card-ord-of-nat* [*simp*]: *Card* (*ord-of-nat*  $n$ )  
*<proof>*

**lemma** *Card-0* [*iff*]: *Card*  $0$   
*<proof>*

**lemma** *CardI*:  $\llbracket Ord\ i; \bigwedge j. \llbracket j < i; Ord\ j \rrbracket \implies \neg elts\ j \approx elts\ i \rrbracket \implies Card\ i$   
*<proof>*

**lemma** *vc*- $0$  [*simp*]:  $vcard\ 0 = 0$   
*<proof>*

**lemma** *Ord-cardinal* [*simp,intro!*]: *Ord*( $vcard\ a$ )  
*<proof>*

**lemma** *gcard-big-0*:  $\neg small\ X \implies gcard\ X = 0$   
*<proof>*

**lemma** *gcard-eq-card*:  
**assumes** *finite*  $X$  **shows**  $gcard\ X = ord-of-nat (card\ X)$   
*<proof>*

**lemma** *gcard-empty-0* [*simp*]:  $gcard\ \{\} = 0$   
*<proof>*

**lemma** *gcard-single-1* [*simp*]:  $gcard\ \{x\} = 1$   
*<proof>*

**lemma** *Card-gcard* [*iff*]: *Card* ( $gcard\ X$ )  
*<proof>*

**lemma** *gcard-epoll*:  $small\ X \implies elts (gcard\ X) \approx X$   
*<proof>*

**lemma** *lepoll-imp-gcard-le*:  
**assumes**  $A \lesssim B$  *small B*  
**shows**  $\text{gcard } A \leq \text{gcard } B$   
 $\langle \text{proof} \rangle$

**lemma** *gcard-image-le*:  
**assumes** *small A* **shows**  $\text{gcard } (f \cdot A) \leq \text{gcard } A$   
 $\langle \text{proof} \rangle$

**lemma** *subset-imp-gcard-le*:  
**assumes**  $A \subseteq B$  *small B*  
**shows**  $\text{gcard } A \leq \text{gcard } B$   
 $\langle \text{proof} \rangle$

**lemma** *gcard-le-lepoll*:  $\llbracket \text{gcard } A \leq \alpha; \text{small } A \rrbracket \implies A \lesssim \text{elts } \alpha$   
 $\langle \text{proof} \rangle$

## 2.12 Cardinality of a set

The cardinals are the initial ordinals.

**lemma** *Card-iff-initial*:  $\text{Card } \kappa \longleftrightarrow \text{Ord } \kappa \wedge (\forall \alpha. \text{Ord } \alpha \wedge \alpha < \kappa \longrightarrow \sim \text{elts } \alpha \approx \text{elts } \kappa)$   
 $\langle \text{proof} \rangle$

**lemma** *Card- $\omega$  [iff]*:  $\text{Card } \omega$   
 $\langle \text{proof} \rangle$

**lemma** *lt-Card-imp-lesspoll*:  $\llbracket i < a; \text{Card } a; \text{Ord } i \rrbracket \implies \text{elts } i \prec \text{elts } a$   
 $\langle \text{proof} \rangle$

**lemma** *lepoll-imp-Card-le*:  
**assumes**  $\text{elts } a \lesssim \text{elts } b$  **shows**  $\text{vcard } a \leq \text{vcard } b$   
 $\langle \text{proof} \rangle$

**lemma** *lepoll-cardinal-le*:  $\llbracket \text{elts } A \lesssim \text{elts } i; \text{Ord } i \rrbracket \implies \text{vcard } A \leq i$   
 $\langle \text{proof} \rangle$

**lemma** *cardinal-le-lepoll*:  $\text{vcard } A \leq \alpha \implies \text{elts } A \lesssim \text{elts } \alpha$   
 $\langle \text{proof} \rangle$

**lemma** *lesspoll-imp-Card-less*:  
**assumes**  $\text{elts } a \prec \text{elts } b$  **shows**  $\text{vcard } a < \text{vcard } b$   
 $\langle \text{proof} \rangle$

**lemma** *Card-Union [simp,intro]*:  
**assumes**  $A: \bigwedge x. x \in A \implies \text{Card}(x)$  **shows**  $\text{Card}(\bigsqcup A)$   
 $\langle \text{proof} \rangle$

**lemma** *Card-UN*:  $(\bigwedge x. x \in A \implies \text{Card}(K x)) \implies \text{Card}(\text{Sup } (K \text{ ` } A))$   
 ⟨proof⟩

## 2.13 Transfinite recursion for definitions based on the three cases of ordinals

**definition**

*transrec3* ::  $[V, [V, V] \Rightarrow V, [V, V \Rightarrow V] \Rightarrow V, V] \Rightarrow V$  **where**  
*transrec3* *a b c*  $\equiv$   
*transrec*  $(\lambda r x.$   
 if  $x=0$  then *a*  
 else if *Limit* *x* then *c*  $x$   $(\lambda y \in \text{elts } x. r y)$   
 else *b*(*pred* *x*)  $(r$  (*pred* *x*)))

**lemma** *transrec3-0* [*simp*]: *transrec3* *a b c* 0 = *a*  
 ⟨proof⟩

**lemma** *transrec3-succ* [*simp*]:  
*transrec3* *a b c* (*succ* *i*) = *b* *i* (*transrec3* *a b c* *i*)  
 ⟨proof⟩

**lemma** *transrec3-Limit* [*simp*]:  
*Limit* *i*  $\implies$  *transrec3* *a b c* *i* = *c* *i*  $(\lambda j \in \text{elts } i. \text{transrec3 } a b c j)$   
 ⟨proof⟩

## 2.14 Cardinal Addition

**definition** *cadd* ::  $[V, V] \Rightarrow V$  (**infixl**  $\langle \oplus \rangle$  65)  
**where**  $\kappa \oplus \mu \equiv \text{vcard } (\kappa \uplus \mu)$

### 2.14.1 Cardinal addition is commutative

**lemma** *vsum-commute-epoll*: *elts*  $(a \uplus b) \approx \text{elts } (b \uplus a)$   
 ⟨proof⟩

**lemma** *cadd-commute*:  $i \oplus j = j \oplus i$   
 ⟨proof⟩

### 2.14.2 Cardinal addition is associative

**lemma** *sum-assoc-bij*:  
*bij-betw*  $(\lambda z \in \text{elts } ((a \uplus b) \uplus c). \text{sum-case}(\text{sum-case } \text{Inl } (\lambda y. \text{Inr}(\text{Inl } y))) (\lambda y. \text{Inr}(\text{Inr } y)) z)$   
 $(\text{elts } ((a \uplus b) \uplus c)) (\text{elts } (a \uplus (b \uplus c)))$   
 ⟨proof⟩

**lemma** *sum-assoc-epoll*: *elts*  $((a \uplus b) \uplus c) \approx \text{elts } (a \uplus (b \uplus c))$   
 ⟨proof⟩

**lemma** *elts-vcadd-vsum-epoll*: *elts*  $(\text{vcadd } (i \uplus j)) \approx \text{Inl } \text{ ` } \text{elts } i \cup \text{Inr } \text{ ` } \text{elts } j$

*<proof>*

**lemma** *cadd-assoc*:  $(i \oplus j) \oplus k = i \oplus (j \oplus k)$   
*<proof>*

**lemma** *cadd-left-commute*:  $j \oplus (i \oplus k) = i \oplus (j \oplus k)$   
*<proof>*

**lemmas** *cadd-ac = cadd-assoc cadd-commute cadd-left-commute*

0 is the identity for addition

**lemma** *vsum-0-epoll*:  $\text{elts } (0 \uplus a) \approx \text{elts } a$   
*<proof>*

**lemma** *cadd-0 [simp]*:  $\text{Card } \kappa \implies 0 \oplus \kappa = \kappa$   
*<proof>*

**lemma** *cadd-0-right [simp]*:  $\text{Card } \kappa \implies \kappa \oplus 0 = \kappa$   
*<proof>*

**lemma** *vsum-lepoll-self*:  $\text{elts } a \lesssim \text{elts } (a \uplus b)$   
*<proof>*

**lemma** *cadd-le-self*:  
**assumes**  $\kappa$ :  $\text{Card } \kappa$  **shows**  $\kappa \leq \kappa \oplus a$   
*<proof>*

Monotonicity of addition

**lemma** *cadd-le-mono*:  $[\kappa' \leq \kappa; \mu' \leq \mu] \implies \kappa' \oplus \mu' \leq \kappa \oplus \mu$   
*<proof>*

## 2.15 Cardinal multiplication

**definition** *cmult* ::  $[V, V] \Rightarrow V$  (**infixl**  $\langle \otimes \rangle$  70)  
**where**  $\kappa \otimes \mu \equiv \text{vcard } (V\text{Sigma } \kappa (\lambda z. \mu))$

### 2.15.1 Cardinal multiplication is commutative

**lemma** *prod-bij*:  $[[\text{bij-betw } f \ A \ C; \text{bij-betw } g \ B \ D]]$   
 $\implies \text{bij-betw } (\lambda(x, y). (f \ x, g \ y)) \ (A \times B) \ (C \times D)$   
*<proof>*

**lemma** *cmult-commute*:  $i \otimes j = j \otimes i$   
*<proof>*

### 2.15.2 Cardinal multiplication is associative

**lemma** *elts-vcard-VSigma-epoll*:  $\text{elts } (\text{vcard } (\text{vtimes } i \ j)) \approx \text{elts } i \times \text{elts } j$   
*<proof>*

**lemma** *elts-cmult*:  $\text{elts } (\kappa' \otimes \kappa) \approx \text{elts } \kappa' \times \text{elts } \kappa$   
 ⟨proof⟩

**lemma** *cmult-assoc*:  $(i \otimes j) \otimes k = i \otimes (j \otimes k)$   
 ⟨proof⟩

### 2.15.3 Cardinal multiplication distributes over addition

**lemma** *cadd-cmult-distrib*:  $(i \oplus j) \otimes k = (i \otimes k) \oplus (j \otimes k)$   
 ⟨proof⟩

Multiplication by 0 yields 0

**lemma** *cmult-0* [*simp*]:  $0 \otimes i = 0$   
 ⟨proof⟩

1 is the identity for multiplication

**lemma** *cmult-1* [*simp*]: **assumes** *Card*  $\kappa$  **shows**  $1 \otimes \kappa = \kappa$   
 ⟨proof⟩

### 2.16 Some inequalities for multiplication

**lemma** *cmult-square-le*: **assumes** *Card*  $\kappa$  **shows**  $\kappa \leq \kappa \otimes \kappa$   
 ⟨proof⟩

Multiplication by a non-empty set

**lemma** *cmult-le-self*: **assumes** *Card*  $\kappa$   $\alpha \neq 0$  **shows**  $\kappa \leq \kappa \otimes \alpha$   
 ⟨proof⟩

Monotonicity of multiplication

**lemma** *cmult-le-mono*:  $\llbracket \kappa' \leq \kappa; \mu' \leq \mu \rrbracket \implies \kappa' \otimes \mu' \leq \kappa \otimes \mu$   
 ⟨proof⟩

**lemma** *vcard-Sup-le-cmult*:

**assumes** *small*  $U$  **and**  $\kappa: \bigwedge x. x \in U \implies \text{vcard } x \leq \kappa$

**shows**  $\text{vcard } (\bigsqcup U) \leq \text{vcard } (\text{set } U) \otimes \kappa$

⟨proof⟩

### 2.17 The finite cardinals

**lemma** *succ-lepoll-succD*:  $\text{elts } (\text{succ}(m)) \lesssim \text{elts } (\text{succ}(n)) \implies \text{elts } m \lesssim \text{elts } n$   
 ⟨proof⟩

Congruence law for *succ* under equipollence

**lemma** *succ-epoll-cong*:  $\text{elts } a \approx \text{elts } b \implies \text{elts } (\text{succ}(a)) \approx \text{elts } (\text{succ}(b))$   
 ⟨proof⟩

**lemma** *sum-succ-epoll*:  $\text{elts } (\text{succ } a \uplus b) \approx \text{elts } (\text{succ}(a \uplus b))$

*<proof>*

**lemma** *cadd-succ*:  $\text{succ } m \oplus n = \text{vcard } (\text{succ}(m \oplus n))$   
*<proof>*

**lemma** *nat-cadd-eq-add*:  $\text{ord-of-nat } m \oplus \text{ord-of-nat } n = \text{ord-of-nat } (m + n)$   
*<proof>*

**lemma** *vcard-disjoint-sup*:  
**assumes**  $x \sqcap y = 0$  **shows**  $\text{vcard } (x \sqcup y) = \text{vcard } x \oplus \text{vcard } y$   
*<proof>*

**lemma** *vcard-sup*:  $\text{vcard } (x \sqcup y) \leq \text{vcard } x \oplus \text{vcard } y$   
*<proof>*

## 2.18 Infinite cardinals

**definition** *InfCard* ::  $V \Rightarrow \text{bool}$   
**where**  $\text{InfCard } \kappa \equiv \text{Card } \kappa \wedge \omega \leq \kappa$

**lemma** *InfCard-iff*:  $\text{InfCard } \kappa \longleftrightarrow \text{Card } \kappa \wedge \text{infinite } (\text{elts } \kappa)$   
*<proof>*

**lemma** *InfCard-ge-ord-of-nat*:  
**assumes**  $\text{InfCard } \kappa$  **shows**  $\text{ord-of-nat } n \leq \kappa$   
*<proof>*

**lemma** *InfCard-not-0[iff]*:  $\neg \text{InfCard } 0$   
*<proof>*

**definition** *csucc* ::  $V \Rightarrow V$   
**where**  $\text{csucc } \kappa \equiv \text{LEAST } \kappa'. \text{Ord } \kappa' \wedge (\text{Card } \kappa' \wedge \kappa < \kappa')$

**lemma** *less-vcard-VPow*:  $\text{vcard } A < \text{vcard } (\text{VPow } A)$   
*<proof>*

**lemma** *greater-Card*:  
**assumes**  $\text{Card } \kappa$  **shows**  $\kappa < \text{vcard } (\text{VPow } \kappa)$   
*<proof>*

**lemma**  
**assumes**  $\text{Card } \kappa$   
**shows**  $\text{Card-csucc } [\text{simp}]: \text{Card } (\text{csucc } \kappa)$  **and**  $\text{less-csucc } [\text{simp}]: \kappa < \text{csucc } \kappa$   
*<proof>*

**lemma** *le-csucc*:  
**assumes**  $\text{Card } \kappa$  **shows**  $\kappa \leq \text{csucc } \kappa$   
*<proof>*

**lemma** *csucc-le*:  $\llbracket \text{Card } \mu; \kappa \in \text{elts } \mu \rrbracket \implies \text{csucc } \kappa \leq \mu$   
*<proof>*

**lemma** *finite-csucc*:  $a \in \text{elts } \omega \implies \text{csucc } a = \text{succ } a$   
*<proof>*

**lemma** *Finite-imp-cardinal-cons* [*simp*]:  
**assumes** *FA*: *finite A* **and** *a*:  $a \notin A$   
**shows**  $\text{vcard } (\text{set } (\text{insert } a A)) = \text{csucc}(\text{vcard } (\text{set } A))$   
*<proof>*

**lemma** *vcard-finite-set*:  $\text{finite } A \implies \text{vcard } (\text{set } A) = \text{ord-of-nat } (\text{card } A)$   
*<proof>*

**lemma** *lt-csucc-iff*:  
**assumes** *Ord*  $\alpha$  *Card*  $\kappa$   
**shows**  $\alpha < \text{csucc } \kappa \longleftrightarrow \text{vcard } \alpha \leq \kappa$   
*<proof>*

**lemma** *Card-lt-csucc-iff*:  $\llbracket \text{Card } \kappa'; \text{Card } \kappa \rrbracket \implies (\kappa' < \text{csucc } \kappa) = (\kappa' \leq \kappa)$   
*<proof>*

**lemma** *csucc-lt-csucc-iff*:  $\llbracket \text{Card } \kappa'; \text{Card } \kappa \rrbracket \implies (\text{csucc } \kappa' < \text{csucc } \kappa) = (\kappa' < \kappa)$   
*<proof>*

**lemma** *csucc-le-csucc-iff*:  $\llbracket \text{Card } \kappa'; \text{Card } \kappa \rrbracket \implies (\text{csucc } \kappa' \leq \text{csucc } \kappa) = (\kappa' \leq \kappa)$   
*<proof>*

**lemma** *csucc-0* [*simp*]:  $\text{csucc } 0 = 1$   
*<proof>*

**lemma** *Card-Un* [*simp,intro*]:  
**assumes** *Card*  $x$  *Card*  $y$  **shows** *Card*( $x \sqcup y$ )  
*<proof>*

**lemma** *InfCard-csucc*:  $\text{InfCard } \kappa \implies \text{InfCard } (\text{csucc } \kappa)$   
*<proof>*

Kunen's Lemma 10.11

**lemma** *InfCard-is-Limit*:  
**assumes** *InfCard*  $\kappa$  **shows** *Limit*  $\kappa$   
*<proof>*

## 2.19 Toward's Kunen's Corollary 10.13 (1)

Kunen's Theorem 10.12

**lemma** *InfCard-csquare-eq*:

**assumes**  $\text{InfCard}(\kappa)$  **shows**  $\kappa \otimes \kappa = \kappa$   
 ⟨proof⟩

**lemma**  $\text{InfCard-le-cmult-eq}$ :  
**assumes**  $\text{InfCard } \kappa \ \mu \leq \kappa \ \mu \neq 0$   
**shows**  $\kappa \otimes \mu = \kappa$   
 ⟨proof⟩

Kunen's Corollary 10.13 (1), for cardinal multiplication

**lemma**  $\text{InfCard-cmult-eq}$ :  $\llbracket \text{InfCard } \kappa; \text{InfCard } \mu \rrbracket \implies \kappa \otimes \mu = \kappa \sqcup \mu$   
 ⟨proof⟩

**lemma**  $\text{cmult-succ}$ :  
 $\text{succ}(m) \otimes n = n \oplus (m \otimes n)$   
 ⟨proof⟩

**lemma**  $\text{cmult-2}$ :  
**assumes**  $\text{Card } n$  **shows**  $\text{ord-of-nat } 2 \otimes n = n \oplus n$   
 ⟨proof⟩

**lemma**  $\text{InfCard-cdouble-eq}$ :  
**assumes**  $\text{InfCard } \kappa$  **shows**  $\kappa \oplus \kappa = \kappa$   
 ⟨proof⟩

Corollary 10.13 (1), for cardinal addition

**lemma**  $\text{InfCard-le-cadd-eq}$ :  $\llbracket \text{InfCard } \kappa; \mu \leq \kappa \rrbracket \implies \kappa \oplus \mu = \kappa$   
 ⟨proof⟩

**lemma**  $\text{InfCard-cadd-eq}$ :  $\llbracket \text{InfCard } \kappa; \text{InfCard } \mu \rrbracket \implies \kappa \oplus \mu = \kappa \sqcup \mu$   
 ⟨proof⟩

**lemma**  $\text{csucc-le-Card-iff}$ :  $\llbracket \text{Card } \kappa'; \text{Card } \kappa \rrbracket \implies \text{csucc } \kappa' \leq \kappa \iff \kappa' < \kappa$   
 ⟨proof⟩

**lemma**  $\text{cadd-InfCard-le}$ :  
**assumes**  $\alpha \leq \kappa \ \beta \leq \kappa \ \text{InfCard } \kappa$   
**shows**  $\alpha \oplus \beta \leq \kappa$   
 ⟨proof⟩

**lemma**  $\text{cmult-InfCard-le}$ :  
**assumes**  $\alpha \leq \kappa \ \beta \leq \kappa \ \text{InfCard } \kappa$   
**shows**  $\alpha \otimes \beta \leq \kappa$   
 ⟨proof⟩

## 2.20 The Aleph-sequence

This is the well-known transfinite enumeration of the cardinal numbers.

**definition**  $\text{Aleph} :: V \Rightarrow V \quad (\aleph \rightarrow) [90] 90)$

**where**  $Aleph \equiv transrec (\lambda f x. \omega \sqcup \bigsqcup ((\lambda y. csucc(f y)) \text{‘} elts x))$

**lemma** *Aleph*:  $Aleph \alpha = \omega \sqcup (\bigsqcup_{y \in elts \alpha} csucc (Aleph y))$   
 $\langle proof \rangle$

**lemma** *InfCard-Aleph* [*simp*, *intro*]:  $InfCard(Aleph x)$   
 $\langle proof \rangle$

**lemma** *Card-Aleph* [*simp*, *intro*]:  $Card(Aleph x)$   
 $\langle proof \rangle$

**lemma** *Aleph-0* [*simp*]:  $\aleph 0 = \omega$   
 $\langle proof \rangle$

**lemma** *mem-Aleph-succ*:  $\aleph \alpha \in elts (Aleph (succ \alpha))$   
 $\langle proof \rangle$

**lemma** *Aleph-lt-succD* [*simp*]:  $\aleph \alpha < Aleph (succ \alpha)$   
 $\langle proof \rangle$

**lemma** *Aleph-succ* [*simp*]:  $Aleph (succ x) = csucc (Aleph x)$   
 $\langle proof \rangle$

**lemma** *csucc-Aleph-le-Aleph*:  $\alpha \in elts \beta \implies csucc (\aleph \alpha) \leq \aleph \beta$   
 $\langle proof \rangle$

**lemma** *Aleph-in-Aleph*:  $\alpha \in elts \beta \implies \aleph \alpha \in elts (\aleph \beta)$   
 $\langle proof \rangle$

**lemma** *Aleph-Limit*:  
**assumes** *Limit*  $\gamma$   
**shows**  $Aleph \gamma = \bigsqcup (Aleph \text{‘} elts \gamma)$   
 $\langle proof \rangle$

**lemma** *Aleph-increasing*:  
**assumes** *ab*:  $\alpha < \beta$  *Ord*  $\alpha$  *Ord*  $\beta$  **shows**  $\aleph \alpha < \aleph \beta$   
 $\langle proof \rangle$

**lemma** *countable-iff-le-Aleph0*:  $countable (elts A) \iff vcard A \leq \aleph 0$   
 $\langle proof \rangle$

**lemma** *Aleph-csquare-eq* [*simp*]:  $\aleph \alpha \otimes \aleph \alpha = \aleph \alpha$   
 $\langle proof \rangle$

**lemma** *vcards-Aleph* [*simp*]:  $vcards (\aleph \alpha) = \aleph \alpha$   
 $\langle proof \rangle$

**lemma** *omega-le-Aleph* [*simp*]:  $\omega \leq \aleph \alpha$   
 $\langle proof \rangle$

**lemma** *finite-iff-less-Aleph0*: *finite* (elts  $x$ )  $\longleftrightarrow$  *vcard*  $x < \omega$   
*<proof>*

**lemma** *countable-iff-vcard-less1*: *countable* (elts  $x$ )  $\longleftrightarrow$  *vcard*  $x < \aleph 1$   
*<proof>*

**lemma** *countable-infinite-vcard*: *countable* (elts  $x$ )  $\wedge$  *infinite* (elts  $x$ )  $\longleftrightarrow$  *vcard*  $x = \aleph 0$   
*<proof>*

## 2.21 The ordinal $\omega 1$

**abbreviation**  $\omega 1 \equiv \text{Aleph } 1$

**lemma** *Ord- $\omega 1$*  [*simp*]: *Ord*  $\omega 1$   
*<proof>*

**lemma** *omega- $\omega 1$*  [*iff*]:  $\omega \in$  elts  $\omega 1$   
*<proof>*

**lemma** *ord-of-nat- $\omega 1$*  [*iff*]: *ord-of-nat*  $n \in$  elts  $\omega 1$   
*<proof>*

**lemma** *countable-iff-less- $\omega 1$* :  
**assumes** *Ord*  $\alpha$   
**shows** *countable* (elts  $\alpha$ )  $\longleftrightarrow$   $\alpha < \omega 1$   
*<proof>*

**lemma** *less- $\omega 1$ -imp-countable*:  
**assumes**  $\alpha \in$  elts  $\omega 1$   
**shows** *countable* (elts  $\alpha$ )  
*<proof>*

**lemma**  *$\omega 1$ -gt0* [*simp*]:  $\omega 1 > 0$   
*<proof>*

**lemma**  *$\omega 1$ -gt1* [*simp*]:  $\omega 1 > 1$   
*<proof>*

**lemma** *Limit- $\omega 1$*  [*simp*]: *Limit*  $\omega 1$   
*<proof>*

**end**

## 3 Addition and Multiplication of Sets

**theory** *Kirby*  
**imports** *ZFC-Cardinals*

**begin**

### 3.1 Generalised Addition

Source: Laurence Kirby, Addition and multiplication of sets Math. Log. Quart. 53, No. 1, 52-65 (2007) / DOI 10.1002/malq.200610026 <http://faculty.baruch.cuny.edu/lkirby/mlqarticlejan2007.pdf>

#### 3.1.1 Addition is a monoid

**instantiation**  $V :: plus$

**begin**

This definition is credited to Tarski

**definition**  $plus-V :: V \Rightarrow V \Rightarrow V$

**where**  $plus-V x \equiv transrec (\lambda f z. x \sqcup set (f \text{ ' } elts z))$

**instance**  $\langle proof \rangle$

**end**

**definition**  $lift :: V \Rightarrow V \Rightarrow V$

**where**  $lift x y \equiv set (plus x \text{ ' } elts y)$

**lemma**  $plus: x + y = x \sqcup set ((+)x \text{ ' } elts y)$

$\langle proof \rangle$

**lemma**  $plus-eq-lift: x + y = x \sqcup lift x y$

$\langle proof \rangle$

Lemma 3.2

**lemma**  $lift-sup-distrib: lift x (a \sqcup b) = lift x a \sqcup lift x b$

$\langle proof \rangle$

**lemma**  $lift-Sup-distrib: small Y \Longrightarrow lift x (\bigsqcup Y) = \bigsqcup (lift x \text{ ' } Y)$

$\langle proof \rangle$

**lemma**  $add-Sup-distrib:$

**fixes**  $x::V$  **shows**  $y \neq 0 \Longrightarrow x + (\bigsqcup z \in elts y. f z) = (\bigsqcup z \in elts y. x + f z)$

$\langle proof \rangle$

**lemma**  $Limit-add-Sup-distrib:$

**fixes**  $x::V$  **shows**  $Limit \alpha \Longrightarrow x + (\bigsqcup z \in elts \alpha. f z) = (\bigsqcup z \in elts \alpha. x + f z)$

$\langle proof \rangle$

Proposition 3.3(ii)

**instantiation**  $V :: monoid-add$

**begin**

**instance**

*<proof>*

**end**

**lemma** *lift-0* [*simp*]: *lift 0 x = x*

*<proof>*

**lemma** *lift-by0* [*simp*]: *lift x 0 = 0*

*<proof>*

**lemma** *lift-by1* [*simp*]: *lift x 1 = set{x}*

*<proof>*

**lemma** *add-eq-0-iff* [*simp*]:

**fixes** *x y*: *V*

**shows**  $x+y = 0 \longleftrightarrow x=0 \wedge y=0$

*<proof>*

**lemma** *plus-vinsert*:  $x + \text{vinsert } z \ y = \text{vinsert } (x+z) \ (x + y)$

*<proof>*

**lemma** *plus-V-succ-right*:  $x + \text{succ } y = \text{succ } (x + y)$

*<proof>*

**lemma** *succ-eq-add1*:  $\text{succ } x = x + 1$

*<proof>*

**lemma** *ord-of-nat-add*:  $\text{ord-of-nat } (m+n) = \text{ord-of-nat } m + \text{ord-of-nat } n$

*<proof>*

**lemma** *succ-0-plus-eq* [*simp*]:

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

**shows**  $\text{succ } 0 + \alpha = \text{succ } \alpha$

*<proof>*

**lemma** *omega-closed-add* [*intro*]:

**assumes**  $\alpha \in \text{elts } \omega \ \beta \in \text{elts } \omega$  **shows**  $\alpha+\beta \in \text{elts } \omega$

*<proof>*

**lemma** *mem-plus-V-E*:

**assumes**  $l \in \text{elts } (x + y)$

**obtains**  $l \in \text{elts } x \mid z$  **where**  $z \in \text{elts } y \ l = x + z$

*<proof>*

**lemma** *not-add-less-right*: **assumes** *Ord y* **shows**  $\neg (x + y < x)$

*<proof>*

**lemma** *not-add-mem-right*:  $\neg (x + y \in \text{elts } x)$

*<proof>*

Proposition 3.3(iii)

**lemma** *add-not-less-TC-self*:  $\neg x + y \sqsubset x$   
*<proof>*

**lemma** *TC-sup-lift*:  $TC\ x \sqcap \text{lift}\ x\ y = 0$   
*<proof>*

**lemma** *lift-lift*:  $\text{lift}\ x\ (\text{lift}\ y\ z) = \text{lift}\ (x+y)\ z$   
*<proof>*

**lemma** *lift-self-disjoint*:  $x \sqcap \text{lift}\ x\ u = 0$   
*<proof>*

**lemma** *sup-lift-eq-lift*:  
**assumes**  $x \sqcup \text{lift}\ x\ u = x \sqcup \text{lift}\ x\ v$   
**shows**  $\text{lift}\ x\ u = \text{lift}\ x\ v$   
*<proof>*

### 3.1.2 Deeper properties of addition

Proposition 3.4(i)

**proposition** *lift-eq-lift*:  $\text{lift}\ x\ y = \text{lift}\ x\ z \implies y = z$   
*<proof>*

**corollary** *inj-lift*: *inj-on* ( $\text{lift}\ x$ )  $A$   
*<proof>*

**corollary** *add-right-cancel [iff]*:  
**fixes**  $x\ y\ z::V$  **shows**  $x+y = x+z \longleftrightarrow y=z$   
*<proof>*

**corollary** *add-mem-right-cancel [iff]*:  
**fixes**  $x\ y\ z::V$  **shows**  $x+y \in \text{elts}\ (x+z) \longleftrightarrow y \in \text{elts}\ z$   
*<proof>*

**corollary** *add-le-cancel-left [iff]*:  
**fixes**  $x\ y\ z::V$  **shows**  $x+y \leq x+z \longleftrightarrow y \leq z$   
*<proof>*

**corollary** *add-less-cancel-left [iff]*:  
**fixes**  $x\ y\ z::V$  **shows**  $x+y < x+z \longleftrightarrow y < z$   
*<proof>*

**corollary** *lift-le-self [simp]*:  $\text{lift}\ x\ y \leq x \longleftrightarrow y = 0$   
*<proof>*

**lemma** *succ-less- $\omega$ -imp*:  $\text{succ}\ x < \omega \implies x < \omega$   
*<proof>*

Proposition 3.5

**lemma** *card-lift*:  $\text{vcard} (\text{lift } x \ y) = \text{vcard } y$   
*<proof>*

**lemma** *eqpoll-lift*:  $\text{elts} (\text{lift } x \ y) \approx \text{elts } y$   
*<proof>*

**lemma** *vcard-add*:  $\text{vcard} (x + y) = \text{vcard } x \oplus \text{vcard } y$   
*<proof>*

**lemma** *countable-add*:  
  **assumes** *countable* (elts *A*) *countable* (elts *B*)  
  **shows** *countable* (elts (*A+B*))  
*<proof>*

Proposition 3.6

**proposition** *TC-add*:  $TC (x + y) = TC \ x \sqcup \text{lift } x \ (TC \ y)$   
*<proof>*

**corollary** *TC-add'*:  $z \sqsubset x + y \longleftrightarrow z \sqsubset x \vee (\exists v. v \sqsubset y \wedge z = x + v)$   
*<proof>*

Corollary 3.7

**corollary** *vcard-TC-add*:  $\text{vcard} (TC (x+y)) = \text{vcard} (TC \ x) \oplus \text{vcard} (TC \ y)$   
*<proof>*

Corollary 3.8

**corollary** *TC-lift*:  
  **assumes**  $y \neq 0$   
  **shows**  $TC (\text{lift } x \ y) = TC \ x \sqcup \text{lift } x \ (TC \ y)$   
*<proof>*

**proposition** *rank-add-distrib*:  $\text{rank} (x+y) = \text{rank } x + \text{rank } y$   
*<proof>*

**lemma** *Ord-add [simp]*:  $\llbracket \text{Ord } x; \text{Ord } y \rrbracket \Longrightarrow \text{Ord} (x+y)$   
*<proof>*

**lemma** *add-Sup-distrib-id*:  $A \neq 0 \Longrightarrow x + \bigsqcup (\text{elts } A) = (\bigsqcup z \in \text{elts } A. x + z)$   
*<proof>*

**lemma** *add-Limit*:  $\text{Limit } \alpha \Longrightarrow x + \alpha = (\bigsqcup z \in \text{elts } \alpha. x + z)$   
*<proof>*

**lemma** *add-le-left*:  
  **assumes** *Ord*  $\alpha$  *Ord*  $\beta$  **shows**  $\beta \leq \alpha + \beta$   
*<proof>*

**lemma** *plus- $\omega$ -equals- $\omega$* :  
**assumes**  $\alpha \in \text{elts } \omega$  **shows**  $\alpha + \omega = \omega$   
 $\langle \text{proof} \rangle$

**lemma** *one-plus- $\omega$ -equals- $\omega$*  [*simp*]:  $1 + \omega = \omega$   
 $\langle \text{proof} \rangle$

### 3.1.3 Cancellation / set subtraction

**definition** *vle* ::  $V \Rightarrow V \Rightarrow \text{bool}$  (**infix**  $\langle \trianglelefteq \rangle$  50)  
**where**  $x \trianglelefteq y \equiv \exists z::V. x+z = y$

**lemma** *vle-refl* [*iff*]:  $x \trianglelefteq x$   
 $\langle \text{proof} \rangle$

**lemma** *vle-antisym*:  $\llbracket x \trianglelefteq y; y \trianglelefteq x \rrbracket \Longrightarrow x = y$   
 $\langle \text{proof} \rangle$

**lemma** *vle-trans* [*trans*]:  $\llbracket x \trianglelefteq y; y \trianglelefteq z \rrbracket \Longrightarrow x \trianglelefteq z$   
 $\langle \text{proof} \rangle$

**definition** *vle-comparable* ::  $V \Rightarrow V \Rightarrow \text{bool}$   
**where** *vle-comparable*  $x y \equiv x \trianglelefteq y \vee y \trianglelefteq x$

Lemma 3.13

**lemma** *comparable*:  
**assumes**  $a+b = c+d$   
**shows** *vle-comparable*  $a c$   
 $\langle \text{proof} \rangle$

**lemma** *vle1*:  $x \trianglelefteq y \Longrightarrow x \leq y$   
 $\langle \text{proof} \rangle$

**lemma** *vle2*:  $x \trianglelefteq y \Longrightarrow x \sqsubseteq y$   
 $\langle \text{proof} \rangle$

**lemma** *vle-iff-le-Ord*:  
**assumes** *Ord*  $\alpha$  *Ord*  $\beta$   
**shows**  $\alpha \trianglelefteq \beta \longleftrightarrow \alpha \leq \beta$   
 $\langle \text{proof} \rangle$

**lemma** *add-le-cancel-left0* [*iff*]:  
**fixes**  $x::V$  **shows**  $x \leq x+z$   
 $\langle \text{proof} \rangle$

**lemma** *add-less-cancel-left0* [*iff*]:  
**fixes**  $x::V$  **shows**  $x < x+z \longleftrightarrow 0 < z$   
 $\langle \text{proof} \rangle$

**lemma** *le-Ord-diff*:

**assumes**  $\alpha \leq \beta$  *Ord*  $\alpha$  *Ord*  $\beta$

**obtains**  $\gamma$  **where**  $\alpha + \gamma = \beta$   $\gamma \leq \beta$  *Ord*  $\gamma$

*<proof>*

**lemma** *plus-Ord-le*:

**assumes**  $\alpha \in \text{elts } \omega$  *Ord*  $\beta$  **shows**  $\alpha + \beta \leq \beta + \alpha$

*<proof>*

**lemma** *add-right-mono*:  $\llbracket \alpha \leq \beta; \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma \rrbracket \implies \alpha + \gamma \leq \beta + \gamma$

*<proof>*

**lemma** *add-strict-mono*:  $\llbracket \alpha < \beta; \gamma < \delta; \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma; \text{Ord } \delta \rrbracket \implies \alpha + \gamma < \beta + \delta$

*<proof>*

**lemma** *add-right-strict-mono*:  $\llbracket \alpha \leq \beta; \gamma < \delta; \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma; \text{Ord } \delta \rrbracket \implies \alpha + \gamma < \beta + \delta$

*<proof>*

**lemma** *Limit-add-Limit* [*simp*]:

**assumes** *Limit*  $\mu$  *Ord*  $\beta$  **shows** *Limit*  $(\beta + \mu)$

*<proof>*

## 3.2 Generalised Difference

**definition** *odiff* **where**  $\text{odiff } y \ x \equiv \text{THE } z::V. (x+z = y) \vee (z=0 \wedge \neg x \leq y)$

**lemma** *vle-imp-odiff-eq*:  $x \leq y \implies x + (\text{odiff } y \ x) = y$

*<proof>*

**lemma** *not-vle-imp-odiff-0*:  $\neg x \leq y \implies (\text{odiff } y \ x) = 0$

*<proof>*

**lemma** *Ord-odiff-eq*:

**assumes**  $\alpha \leq \beta$  *Ord*  $\alpha$  *Ord*  $\beta$

**shows**  $\alpha + \text{odiff } \beta \ \alpha = \beta$

*<proof>*

**lemma** *Ord-odiff*:

**assumes** *Ord*  $\alpha$  *Ord*  $\beta$  **shows** *Ord*  $(\text{odiff } \beta \ \alpha)$

*<proof>*

**lemma** *Ord-odiff-le*:

**assumes** *Ord*  $\alpha$  *Ord*  $\beta$  **shows**  $\text{odiff } \beta \ \alpha \leq \beta$

*<proof>*

**lemma** *odiff-0-right* [*simp*]:  $\text{odiff } x \ 0 = x$

*<proof>*

**lemma** *odiff-succ*:  $y \trianglelefteq x \implies \text{odiff } (\text{succ } x) y = \text{succ } (\text{odiff } x y)$   
*<proof>*

**lemma** *odiff-eq-iff*:  $z \trianglelefteq x \implies \text{odiff } x z = y \longleftrightarrow x = z + y$   
*<proof>*

**lemma** *odiff-le-iff*:  $z \trianglelefteq x \implies \text{odiff } x z \leq y \longleftrightarrow x \leq z + y$   
*<proof>*

**lemma** *odiff-less-iff*:  $z \trianglelefteq x \implies \text{odiff } x z < y \longleftrightarrow x < z + y$   
*<proof>*

**lemma** *odiff-ge-iff*:  $z \trianglelefteq x \implies \text{odiff } x z \geq y \longleftrightarrow x \geq z + y$   
*<proof>*

**lemma** *Ord-odiff-le-iff*:  $\llbracket \alpha \leq x; \text{Ord } x; \text{Ord } \alpha \rrbracket \implies \text{odiff } x \alpha \leq y \longleftrightarrow x \leq \alpha + y$   
*<proof>*

**lemma** *odiff-le-odiff*:  
**assumes**  $x \trianglelefteq y$  **shows**  $\text{odiff } x z \leq \text{odiff } y z$   
*<proof>*

**lemma** *Ord-odiff-le-odiff*:  $\llbracket x \leq y; \text{Ord } x; \text{Ord } y \rrbracket \implies \text{odiff } x \alpha \leq \text{odiff } y \alpha$   
*<proof>*

**lemma** *Ord-odiff-less-odiff*:  $\llbracket \alpha \leq x; x < y; \text{Ord } x; \text{Ord } y; \text{Ord } \alpha \rrbracket \implies \text{odiff } x \alpha < \text{odiff } y \alpha$   
*<proof>*

**lemma** *Ord-odiff-less-imp-less*:  $\llbracket \text{odiff } x \alpha < \text{odiff } y \alpha; \text{Ord } x; \text{Ord } y \rrbracket \implies x < y$   
*<proof>*

**lemma** *odiff-add-cancel [simp]*:  $\text{odiff } (x + y) x = y$   
*<proof>*

**lemma** *odiff-add-cancel-0 [simp]*:  $\text{odiff } x x = 0$   
*<proof>*

**lemma** *odiff-add-cancel-both [simp]*:  $\text{odiff } (x + y) (x + z) = \text{odiff } y z$   
*<proof>*

### 3.3 Generalised Multiplication

Credited to Dana Scott

**instantiation**  $V :: \text{times}$   
**begin**

This definition is credited to Tarski

**definition** *times-V* ::  $V \Rightarrow V \Rightarrow V$   
**where** *times-V*  $x \equiv \text{transrec } (\lambda f y. \sqcup ((\lambda u. \text{lift } (f u) x) \text{ ' } \text{elts } y))$

**instance**  $\langle \text{proof} \rangle$   
**end**

**lemma** *mult*:  $x * y = (\sqcup u \in \text{elts } y. \text{lift } (x * u) x)$   
 $\langle \text{proof} \rangle$

**lemma** *elts-multE*:  
**assumes**  $z \in \text{elts } (x * y)$   
**obtains**  $u v$  **where**  $u \in \text{elts } x \ v \in \text{elts } y \ z = x*v + u$   
 $\langle \text{proof} \rangle$

Lemma 4.2

**lemma** *mult-zero-right* [*simp*]:  
**fixes**  $x::V$  **shows**  $x * 0 = 0$   
 $\langle \text{proof} \rangle$

**lemma** *mult-insert*:  $x * (\text{vinsert } y z) = x*z \sqcup \text{lift } (x*y) x$   
 $\langle \text{proof} \rangle$

**lemma** *mult-succ*:  $x * \text{succ } y = x*y + x$   
 $\langle \text{proof} \rangle$

**lemma** *ord-of-nat-mult*:  $\text{ord-of-nat } (m*n) = \text{ord-of-nat } m * \text{ord-of-nat } n$   
 $\langle \text{proof} \rangle$

**lemma** *omega-closed-mult* [*intro*]:  
**assumes**  $\alpha \in \text{elts } \omega \ \beta \in \text{elts } \omega$  **shows**  $\alpha*\beta \in \text{elts } \omega$   
 $\langle \text{proof} \rangle$

**lemma** *zero-imp-le-mult*:  $0 \in \text{elts } y \implies x \leq x*y$   
 $\langle \text{proof} \rangle$

### 3.3.1 Proposition 4.3

**lemma** *mult-zero-left* [*simp*]:  
**fixes**  $x::V$  **shows**  $0 * x = 0$   
 $\langle \text{proof} \rangle$

**lemma** *mult-sup-distrib*:  
**fixes**  $x::V$  **shows**  $x * (y \sqcup z) = x*y \sqcup x*z$   
 $\langle \text{proof} \rangle$

**lemma** *mult-Sup-distrib*:  $\text{small } Y \implies x * (\sqcup Y) = \sqcup ((*) x \text{ ' } Y)$  **for**  $Y::V \text{ set}$   
 $\langle \text{proof} \rangle$

**lemma** *mult-lift-imp-distrib*:  $x * (\text{lift } y \ z) = \text{lift } (x*y) \ (x*z) \implies x * (y+z) = x*y + x*z$   
 <proof>

**lemma** *mult-lift*:  $x * (\text{lift } y \ z) = \text{lift } (x*y) \ (x*z)$   
 <proof>

**lemma** *mult-Limit*:  $\text{Limit } \gamma \implies x * \gamma = \bigsqcup ((*) \ x \ \text{elts } \gamma)$   
 <proof>

**lemma** *add-mult-distrib*:  $x * (y+z) = x*y + x*z$  **for**  $x::V$   
 <proof>

**instantiation**  $V :: \text{monoid-mult}$

**begin**

**instance**

<proof>

**end**

**lemma** *le-mult*:  
**assumes**  $\text{Ord } \beta \ \beta \neq 0$  **shows**  $\alpha \leq \alpha * \beta$   
 <proof>

**lemma** *mult-sing-1* [*simp*]:  
**fixes**  $x::V$  **shows**  $x * \text{set}\{1\} = \text{lift } x \ x$   
 <proof>

**lemma** *mult-2-right* [*simp*]:  
**fixes**  $x::V$  **shows**  $x * \text{set}\{0,1\} = x+x$   
 <proof>

**lemma** *Ord-mult* [*simp*]:  $[\text{Ord } y; \text{Ord } x] \implies \text{Ord } (x*y)$   
 <proof>

### 3.3.2 Proposition 4.4-5

**proposition** *rank-mult-distrib*:  $\text{rank } (x*y) = \text{rank } x * \text{rank } y$   
 <proof>

**lemma** *mult-le1*:  
**fixes**  $y::V$  **assumes**  $y \neq 0$  **shows**  $x \sqsubseteq x * y$   
 <proof>

**lemma** *mult-eq-0-iff* [*simp*]:  
**fixes**  $y::V$  **shows**  $x * y = 0 \iff x=0 \vee y=0$   
 <proof>

**lemma** *lift-lemma*:

**assumes**  $x \neq 0 \ y \neq 0$  **shows**  $\neg \text{lift } (x * y) \ x \leq x$   
*<proof>*

**lemma** *mult-le2*:

**fixes**  $y::V$  **assumes**  $x \neq 0 \ y \neq 0 \ y \neq 1$  **shows**  $x \sqsubset x * y$   
*<proof>*

**lemma** *elts-mult- $\omega E$* :

**assumes**  $x \in \text{elts } (y * \omega)$   
**obtains**  $n$  **where**  $n \neq 0 \ x \in \text{elts } (y * \text{ord-of-nat } n) \wedge m. m < n \implies x \notin \text{elts } (y * \text{ord-of-nat } m)$   
*<proof>*

### 3.3.3 Theorem 4.6

**theorem** *mult-eq-imp-0*:

**assumes**  $a*x = a*y + b \ b \sqsubset a$   
**shows**  $b=0$   
*<proof>*

### 3.3.4 Theorem 4.7

**lemma** *mult-cancellation-half*:

**assumes**  $a*x + r \leq a*y + s \ r \sqsubset a \ s \sqsubset a$   
**shows**  $x \leq y$   
*<proof>*

**theorem** *mult-cancellation-lemma*:

**assumes**  $a*x + r = a*y + s \ r \sqsubset a \ s \sqsubset a$   
**shows**  $x=y \wedge r=s$   
*<proof>*

**corollary** *mult-cancellation [simp]*:

**fixes**  $a::V$   
**assumes**  $a \neq 0$   
**shows**  $a*x = a*y \longleftrightarrow x=y$   
*<proof>*

**corollary** *mult-cancellation-less*:

**assumes**  $lt: a*x + r < a*y + s$  **and**  $r \sqsubset a \ s \sqsubset a$   
**obtains**  $x < y \mid x = y \ r < s$   
*<proof>*

**corollary** *lift-mult-TC-disjoint*:

**fixes**  $x::V$   
**assumes**  $x \neq y$   
**shows**  $\text{lift } (a*x) \ (TC \ a) \sqcap \text{lift } (a*y) \ (TC \ a) = 0$   
*<proof>*

**corollary** *lift-mult-disjoint*:

**fixes**  $x::V$   
**assumes**  $x \neq y$   
**shows**  $\text{lift } (a*x) a \sqcap \text{lift } (a*y) a = 0$   
 $\langle \text{proof} \rangle$

**lemma** *mult-add-mem*:  
**assumes**  $a*x + r \in \text{elts } (a*y) r \sqsubset a$   
**shows**  $x \in \text{elts } y r \in \text{elts } a$   
 $\langle \text{proof} \rangle$

**lemma** *mult-add-mem-0* [simp]:  $a*x \in \text{elts } (a*y) \longleftrightarrow x \in \text{elts } y \wedge 0 \in \text{elts } a$   
 $\langle \text{proof} \rangle$

**lemma** *zero-mem-mult-iff*:  $0 \in \text{elts } (x*y) \longleftrightarrow 0 \in \text{elts } x \wedge 0 \in \text{elts } y$   
 $\langle \text{proof} \rangle$

**lemma** *zero-less-mult-iff* [simp]:  $0 < x*y \longleftrightarrow 0 < x \wedge 0 < y$  **if**  $\text{Ord } x$   
 $\langle \text{proof} \rangle$

**lemma** *mult-cancel-less-iff* [simp]:  
 $\llbracket \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma \rrbracket \implies \alpha*\beta < \alpha*\gamma \longleftrightarrow \beta < \gamma \wedge 0 < \alpha$   
 $\langle \text{proof} \rangle$

**lemma** *mult-cancel-le-iff* [simp]:  
 $\llbracket \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma \rrbracket \implies \alpha*\beta \leq \alpha*\gamma \longleftrightarrow \beta \leq \gamma \vee \alpha=0$   
 $\langle \text{proof} \rangle$

**lemma** *mult-Suc-add-less*:  $\llbracket \alpha < \gamma; \beta < \gamma; \text{Ord } \alpha; \text{Ord } \beta; \text{Ord } \gamma \rrbracket \implies \gamma * \text{ord-of-nat } m + \alpha < \gamma * \text{ord-of-nat } (\text{Suc } m) + \beta$   
 $\langle \text{proof} \rangle$

**lemma** *mult-nat-less-add-less*:  
**assumes**  $m < n \alpha < \gamma \beta < \gamma$  **and**  $\text{ord}: \text{Ord } \alpha \text{ Ord } \beta \text{ Ord } \gamma$   
**shows**  $\gamma * \text{ord-of-nat } m + \alpha < \gamma * \text{ord-of-nat } n + \beta$   
 $\langle \text{proof} \rangle$

**lemma** *add-mult-less-add-mult*:  
**assumes**  $x < y x \in \text{elts } \beta y \in \text{elts } \beta \mu \in \text{elts } \alpha \nu \in \text{elts } \alpha \text{ Ord } \alpha \text{ Ord } \beta$   
**shows**  $\alpha*x + \mu < \alpha*y + \nu$   
 $\langle \text{proof} \rangle$

**lemma** *add-mult-less*:  
**assumes**  $\gamma \in \text{elts } \alpha \nu \in \text{elts } \beta \text{ Ord } \alpha \text{ Ord } \beta$   
**shows**  $\alpha * \nu + \gamma \in \text{elts } (\alpha * \beta)$   
 $\langle \text{proof} \rangle$

**lemma** *Ord-add-mult-iff*:  
**assumes**  $\beta \in \text{elts } \gamma \beta' \in \text{elts } \gamma \text{ Ord } \alpha \text{ Ord } \alpha' \text{ Ord } \gamma$   
**shows**  $\gamma * \alpha + \beta \in \text{elts } (\gamma * \alpha' + \beta') \longleftrightarrow \alpha \in \text{elts } \alpha' \vee \alpha = \alpha' \wedge \beta \in \text{elts } \beta'$

(**is** ?lhs  $\longleftrightarrow$  ?rhs)  
 <proof>

**lemma** *vcard-mult*:  $vcard (x * y) = vcard x \otimes vcard y$   
 <proof>

**proposition** *TC-mult*:  $TC(x * y) = (\bigsqcup r \in elts (TC x). \bigsqcup u \in elts (TC y). set\{x * u + r\})$   
 <proof>

**corollary** *vcard-TC-mult*:  $vcard (TC(x * y)) = vcard (TC x) \otimes vcard (TC y)$   
 <proof>

**lemma** *countable-mult*:  
**assumes** *countable* (elts A) *countable* (elts B)  
**shows** *countable* (elts (A\*B))  
 <proof>

### 3.4 Ordertype properties

**lemma** *ordertype-image-plus*:  
**assumes** *Ord*  $\alpha$   
**shows** *ordertype* ((+) u ‘ elts  $\alpha$ ) *VWF* =  $\alpha$   
 <proof>

**lemma** *ordertype-diff*:  
**assumes**  $\beta + \delta = \alpha$  **and**  $\alpha: \delta \in elts \alpha$  *Ord*  $\alpha$   
**shows** *ordertype* (elts  $\alpha - elts \beta$ ) *VWF* =  $\delta$   
 <proof>

**lemma** *ordertype-interval-eq*:  
**assumes**  $\alpha: Ord \alpha$  **and**  $\beta: Ord \beta$   
**shows** *ordertype* ( $\{\alpha ..< \alpha + \beta\} \cap ON$ ) *VWF* =  $\beta$   
 <proof>

**lemma** *ordertype-Times*:  
**assumes** *small* A *small* B **and** *r*: *wf* r *trans* r *total-on* A r **and** *s*: *wf* s *trans* s *total-on* B s  
**shows** *ordertype* (A\*B) (*r* <\*lex\*> *s*) = *ordertype* B s \* *ordertype* A r (**is** - = ? $\beta$  \* ? $\alpha$ )  
 <proof>

**end**

## 4 Exponentiation of ordinals

**theory** *Ordinal-Exp*  
**imports** *Kirby*



Lemma 3.19

**lemma** *le-oexp*:

**assumes** *Ord*  $\alpha$  *Ord*  $\beta$   $\beta \neq 0$  **shows**  $\alpha \leq \alpha \uparrow \beta$

*<proof>*

Lemma 3.20

**lemma** *le-oexp'*:

**assumes** *Ord*  $\alpha$   $1 < \alpha$  *Ord*  $\beta$  **shows**  $\beta \leq \alpha \uparrow \beta$

*<proof>*

**lemma** *oexp-Limit-le*:

**assumes**  $\beta < \gamma$  *Limit*  $\gamma$  *Ord*  $\beta$   $\alpha > 0$  **shows**  $\alpha \uparrow \beta \leq \alpha \uparrow \gamma$

*<proof>*

**proposition** *oexp-less*:

**assumes**  $\beta: \beta \in \text{elts } \gamma$  **and** *Ord*  $\gamma$  **and**  $\alpha: \alpha > 1$  *Ord*  $\alpha$  **shows**  $\alpha \uparrow \beta < \alpha \uparrow \gamma$

*<proof>*

**corollary** *oexp-less-iff*:

**assumes**  $\alpha > 0$  *Ord*  $\alpha$  *Ord*  $\beta$  *Ord*  $\gamma$  **shows**  $\alpha \uparrow \beta < \alpha \uparrow \gamma \iff \beta \in \text{elts } \gamma \wedge \alpha > 1$

*<proof>*

**lemma**  *$\omega$ -oexp-iff* [*simp*]:  $\llbracket \text{Ord } \alpha; \text{Ord } \beta \rrbracket \implies \omega \uparrow \alpha = \omega \uparrow \beta \iff \alpha = \beta$

*<proof>*

**lemma** *Limit-oexp*:

**assumes** *Limit*  $\gamma$  *Ord*  $\alpha$   $\alpha > 1$  **shows** *Limit*  $(\alpha \uparrow \gamma)$

*<proof>*

**lemma** *oexp-mono*:

**assumes**  $\alpha: \text{Ord } \alpha$   $\alpha \neq 0$  **and**  $\beta: \text{Ord } \beta$   $\gamma \sqsubseteq \beta$  **shows**  $\alpha \uparrow \gamma \leq \alpha \uparrow \beta$

*<proof>*

**lemma** *oexp-mono-le*:

**assumes**  $\gamma \leq \beta$   $\alpha \neq 0$  *Ord*  $\alpha$  *Ord*  $\beta$  *Ord*  $\gamma$  **shows**  $\alpha \uparrow \gamma \leq \alpha \uparrow \beta$

*<proof>*

**lemma** *oexp-sup*:

**assumes**  $\alpha \neq 0$  *Ord*  $\alpha$  *Ord*  $\beta$  *Ord*  $\gamma$  **shows**  $\alpha \uparrow (\beta \sqcup \gamma) = \alpha \uparrow \beta \sqcup \alpha \uparrow \gamma$

*<proof>*

**lemma** *oexp-Sup*:

**assumes**  $\alpha \neq 0$  *Ord*  $\alpha$  **and**  $X: X \subseteq \text{ON small } X$   $X \neq \{\}$  **shows**  $\alpha \uparrow \bigsqcup X =$

$\bigsqcup ((\uparrow) \alpha \text{ ` } X)$

*<proof>*

**lemma** *omega-le-Limit*:  
**assumes** *Limit*  $\mu$  **shows**  $\omega \leq \mu$   
 $\langle$ *proof* $\rangle$

**lemma** *finite-omega-power* [*simp*]:  
**assumes**  $1 < n$   $n \in \text{elts } \omega$  **shows**  $n \uparrow \omega = \omega$   
 $\langle$ *proof* $\rangle$

**proposition** *oexp-add*:  
**assumes** *Ord*  $\alpha$  *Ord*  $\beta$  *Ord*  $\gamma$  **shows**  $\alpha \uparrow (\beta + \gamma) = \alpha \uparrow \beta * \alpha \uparrow \gamma$   
 $\langle$ *proof* $\rangle$

**proposition** *oexp-mult*:  
**assumes** *Ord*  $\alpha$  *Ord*  $\beta$  *Ord*  $\gamma$  **shows**  $\alpha \uparrow (\beta * \gamma) = (\alpha \uparrow \beta) \uparrow \gamma$   
 $\langle$ *proof* $\rangle$

**lemma** *Limit-omega-oexp*:  
**assumes** *Ord*  $\delta$   $\delta \neq 0$   
**shows** *Limit*  $(\omega \uparrow \delta)$   
 $\langle$ *proof* $\rangle$

**lemma** *oexp-mult-commute*:  
**fixes**  $j::\text{nat}$   
**assumes** *Ord*  $\alpha$   
**shows**  $(\alpha \uparrow j) * \alpha = \alpha * (\alpha \uparrow j)$   
 $\langle$ *proof* $\rangle$

**lemma** *oexp- $\omega$ -Limit*: *Limit*  $\beta \implies \omega \uparrow \beta = (\bigsqcup \xi \in \text{elts } \beta. \omega \uparrow \xi)$   
 $\langle$ *proof* $\rangle$

**lemma**  *$\omega$ -power-succ-gtr*: *Ord*  $\alpha \implies \omega \uparrow \alpha * \text{ord-of-nat } n < \omega \uparrow \text{succ } \alpha$   
 $\langle$ *proof* $\rangle$

**lemma** *countable-oexp*:  
**assumes**  $\nu: \alpha \in \text{elts } \omega 1$   
**shows**  $\omega \uparrow \alpha \in \text{elts } \omega 1$   
 $\langle$ *proof* $\rangle$

**end**

## 5 Cantor Normal Form

**theory** *Cantor-NF*  
**imports** *Ordinal-Exp*  
**begin**

## 5.1 Cantor normal form

Lemma 5.1

**lemma** *cnf-1*:

**assumes**  $\alpha: \alpha \in \text{elts } \beta \text{ Ord } \beta$  **and**  $m > 0$

**shows**  $\omega \uparrow \alpha * \text{ord-of-nat } n < \omega \uparrow \beta * \text{ord-of-nat } m$

*<proof>*

**fun** *Cantor-sum* **where**

*Cantor-sum-Nil*:  $\text{Cantor-sum } [] \text{ } ms = 0$

| *Cantor-sum-Nil2*:  $\text{Cantor-sum } (\alpha \# \alpha s) [] = 0$

| *Cantor-sum-Cons*:  $\text{Cantor-sum } (\alpha \# \alpha s) (m \# ms) = (\omega \uparrow \alpha) * \text{ord-of-nat } m + \text{Cantor-sum } \alpha s \text{ } ms$

**abbreviation** *Cantor-dec* ::  $V \text{ list} \Rightarrow \text{bool}$  **where**

*Cantor-dec*  $\equiv \text{sorted-wrt } (>)$

**lemma** *Ord-Cantor-sum*:

**assumes**  $\text{List.set } \alpha s \subseteq \text{ON}$

**shows**  $\text{Ord } (\text{Cantor-sum } \alpha s \text{ } ms)$

*<proof>*

**lemma** *Cantor-dec-Cons-iff* [*simp*]:  $\text{Cantor-dec } (\alpha \# \beta \# \beta s) \longleftrightarrow \beta < \alpha \wedge \text{Cantor-dec } (\beta \# \beta s)$

*<proof>*

Lemma 5.2. The second and third premises aren't really necessary, but their removal requires quite a lot of work.

**lemma** *cnf-2*:

**assumes**  $\text{List.set } (\alpha \# \alpha s) \subseteq \text{ON}$   $\text{list.set } ms \subseteq \{0 < ..\}$   $\text{length } \alpha s = \text{length } ms$

**and**  $\text{Cantor-dec } (\alpha \# \alpha s)$

**shows**  $\omega \uparrow \alpha > \text{Cantor-sum } \alpha s \text{ } ms$

*<proof>*

**proposition** *Cantor-nf-exists*:

**assumes**  $\text{Ord } \alpha$

**obtains**  $\alpha s \text{ } ms$  **where**  $\text{List.set } \alpha s \subseteq \text{ON}$   $\text{list.set } ms \subseteq \{0 < ..\}$   $\text{length } \alpha s = \text{length } ms$

**and**  $\text{Cantor-dec } \alpha s$

**and**  $\alpha = \text{Cantor-sum } \alpha s \text{ } ms$

*<proof>*

**lemma** *Cantor-sum-0E*:

**assumes**  $\text{Cantor-sum } \alpha s \text{ } ms = 0$   $\text{List.set } \alpha s \subseteq \text{ON}$   $\text{list.set } ms \subseteq \{0 < ..\}$   $\text{length } \alpha s = \text{length } ms$

**shows**  $\alpha s = []$

*<proof>*

**lemma** *Cantor-nf-unique-aux*:  
**assumes** *Ord*  $\alpha$   
**and**  $\alpha s ON$ : *List.set*  $\alpha s \subseteq ON$   
**and**  $\beta s ON$ : *List.set*  $\beta s \subseteq ON$   
**and**  $ms$ : *list.set*  $ms \subseteq \{0<..\}$   
**and**  $ns$ : *list.set*  $ns \subseteq \{0<..\}$   
**and**  $mseq$ : *length*  $\alpha s = \text{length } ms$   
**and**  $nseq$ : *length*  $\beta s = \text{length } ns$   
**and**  $\alpha sdec$ : *Cantor-dec*  $\alpha s$   
**and**  $\beta sdec$ : *Cantor-dec*  $\beta s$   
**and**  $\alpha seq$ :  $\alpha = \text{Cantor-sum } \alpha s ms$   
**and**  $\beta seq$ :  $\alpha = \text{Cantor-sum } \beta s ns$   
**shows**  $\alpha s = \beta s \wedge ms = ns$   
*<proof>*

**proposition** *Cantor-nf-unique*:  
**assumes** *Cantor-sum*  $\alpha s ms = \text{Cantor-sum } \beta s ns$   
**and**  $\alpha s ON$ : *List.set*  $\alpha s \subseteq ON$   
**and**  $\beta s ON$ : *List.set*  $\beta s \subseteq ON$   
**and**  $ms$ : *list.set*  $ms \subseteq \{0<..\}$   
**and**  $ns$ : *list.set*  $ns \subseteq \{0<..\}$   
**and**  $mseq$ : *length*  $\alpha s = \text{length } ms$   
**and**  $nseq$ : *length*  $\beta s = \text{length } ns$   
**and**  $\alpha sdec$ : *Cantor-dec*  $\alpha s$   
**and**  $\beta sdec$ : *Cantor-dec*  $\beta s$   
**shows**  $\alpha s = \beta s \wedge ms = ns$   
*<proof>*

**lemma** *less- $\omega$ -power*:  
**assumes** *Ord*  $\alpha 1$  *Ord*  $\beta$   
**and**  $\alpha 2$ :  $\alpha 2 \in \text{elts } \alpha 1$  **and**  $\beta$ :  $\beta < \omega \uparrow \alpha 2$   
**and**  $m 1 > 0$   $m 2 > 0$   
**shows**  $\omega \uparrow \alpha 2 * \text{ord-of-nat } m 2 + \beta < \omega \uparrow \alpha 1 * \text{ord-of-nat } m 1 + (\omega \uparrow \alpha 2 * \text{ord-of-nat } m 2 + \beta)$   
*(is ?lhs < ?rhs)*  
*<proof>*

**lemma** *Cantor-sum-ge*:  
**assumes** *List.set*  $(\alpha \# \alpha s) \subseteq ON$  *list.set*  $ms \subseteq \{0<..\}$  *length*  $ms > 0$   
**shows**  $\omega \uparrow \alpha \leq \text{Cantor-sum } (\alpha \# \alpha s) ms$   
*<proof>*

## 5.2 Simplified Cantor normal form

No coefficients, and the exponents decreasing non-strictly

**fun**  $\omega$ -sum **where**

$\omega$ -sum-Nil:  $\omega$ -sum  $[] = 0$   
|  $\omega$ -sum-Cons:  $\omega$ -sum  $(\alpha\#\alpha s) = (\omega\uparrow\alpha) + \omega$ -sum  $\alpha s$

**abbreviation**  $\omega$ -dec ::  $V$  list  $\Rightarrow$  bool **where**

$\omega$ -dec  $\equiv$  sorted-wrt  $(\geq)$

**lemma** Ord- $\omega$ -sum [simp]: List.set  $\alpha s \subseteq ON \implies$  Ord  $(\omega$ -sum  $\alpha s)$   
(proof)

**lemma**  $\omega$ -dec-Cons-iff [simp]:  $\omega$ -dec  $(\alpha\#\beta\#\beta s) \longleftrightarrow \beta \leq \alpha \wedge \omega$ -dec  $(\beta\#\beta s)$   
(proof)

**lemma**  $\omega$ -sum-0E:

**assumes**  $\omega$ -sum  $\alpha s = 0$  List.set  $\alpha s \subseteq ON$   
**shows**  $\alpha s = []$   
(proof)

**fun**  $\omega$ -of-Cantor **where**

$\omega$ -of-Cantor-Nil:  $\omega$ -of-Cantor  $[]$   $ms = []$   
|  $\omega$ -of-Cantor-Nil2:  $\omega$ -of-Cantor  $(\alpha\#\alpha s)$   $[] = []$   
|  $\omega$ -of-Cantor-Cons:  $\omega$ -of-Cantor  $(\alpha\#\alpha s)$   $(m\#ms) =$  replicate  $m$   $\alpha$  @  $\omega$ -of-Cantor  
 $\alpha s$   $ms$

**lemma**  $\omega$ -sum-append [simp]:  $\omega$ -sum  $(xs$  @  $ys) = \omega$ -sum  $xs + \omega$ -sum  $ys$   
(proof)

**lemma**  $\omega$ -sum-replicate [simp]:  $\omega$ -sum  $($ replicate  $m$   $a) = \omega$   $\uparrow$   $a *$  ord-of-nat  $m$   
(proof)

**lemma**  $\omega$ -sum-of-Cantor [simp]:  $\omega$ -sum  $(\omega$ -of-Cantor  $\alpha s$   $ms) =$  Cantor-sum  $\alpha s$   
 $ms$   
(proof)

**lemma**  $\omega$ -of-Cantor-subset: List.set  $(\omega$ -of-Cantor  $\alpha s$   $ms) \subseteq$  List.set  $\alpha s$   
(proof)

**lemma**  $\omega$ -dec-replicate:  $\omega$ -dec  $($ replicate  $m$   $\alpha$  @  $\alpha s) =$  (if  $m=0$  then  $\omega$ -dec  $\alpha s$  else  
 $\omega$ -dec  $(\alpha\#\alpha s)$ )  
(proof)

**lemma**  $\omega$ -dec-of-Cantor-aux:

**assumes** Cantor-dec  $(\alpha\#\alpha s)$  length  $\alpha s =$  length  $ms$   
**shows**  $\omega$ -dec  $(\omega$ -of-Cantor  $(\alpha\#\alpha s)$   $(m\#ms))$   
(proof)

**lemma**  $\omega$ -dec-of-Cantor:

**assumes**  $Cantor\text{-}dec\ \alpha s$   $length\ \alpha s = length\ ms$

**shows**  $\omega\text{-}dec\ (\omega\text{-of-Cantor}\ \alpha s\ ms)$

$\langle proof \rangle$

**proposition**  $\omega$ -nf-exists:

**assumes**  $Ord\ \alpha$

**obtains**  $\alpha s$  **where**  $List.set\ \alpha s \subseteq ON$  **and**  $\omega\text{-}dec\ \alpha s$  **and**  $\alpha = \omega\text{-}sum\ \alpha s$

$\langle proof \rangle$

**lemma**  $\omega$ -sum-take-drop:  $\omega\text{-}sum\ \alpha s = \omega\text{-}sum\ (take\ k\ \alpha s) + \omega\text{-}sum\ (drop\ k\ \alpha s)$

$\langle proof \rangle$

**lemma**  $in\text{-}elts\text{-}\omega\text{-}sum$ :

**assumes**  $\delta \in elts\ (\omega\text{-}sum\ \alpha s)$

**shows**  $\exists k < length\ \alpha s. \exists \gamma \in elts\ (\omega \uparrow (\alpha s!k)). \delta = \omega\text{-}sum\ (take\ k\ \alpha s) + \gamma$

$\langle proof \rangle$

**lemma**  $\omega$ -le- $\omega$ -sum:  $\llbracket k < length\ \alpha s; List.set\ \alpha s \subseteq ON \rrbracket \implies \omega \uparrow (\alpha s!k) \leq \omega\text{-}sum\ \alpha s$

$\langle proof \rangle$

**lemma**  $\omega$ -sum-less-self:

**assumes**  $List.set\ (\alpha \# \alpha s) \subseteq ON$  **and**  $\omega\text{-}dec\ (\alpha \# \alpha s)$

**shows**  $\omega\text{-}sum\ \alpha s < \omega \uparrow \alpha + \omega\text{-}sum\ \alpha s$

$\langle proof \rangle$

Something like Lemma 5.2 for  $\omega$ -sum

**lemma**  $\omega$ -sum-less- $\omega$ -power:

**assumes**  $\omega\text{-}dec\ (\alpha \# \alpha s)$   $List.set\ (\alpha \# \alpha s) \subseteq ON$

**shows**  $\omega\text{-}sum\ \alpha s < \omega \uparrow \alpha * \omega$

$\langle proof \rangle$

**lemma**  $\omega$ -sum-nf-unique-aux:

**assumes**  $Ord\ \alpha$

**and**  $\alpha s ON: List.set\ \alpha s \subseteq ON$

**and**  $\beta s ON: List.set\ \beta s \subseteq ON$

**and**  $\alpha sdec: \omega\text{-}dec\ \alpha s$

**and**  $\beta sdec: \omega\text{-}dec\ \beta s$

**and**  $\alpha seq: \alpha = \omega\text{-}sum\ \alpha s$

**and**  $\beta seq: \alpha = \omega\text{-}sum\ \beta s$

**shows**  $\alpha s = \beta s$

$\langle proof \rangle$

### 5.3 Indecomposable ordinals

Cf exercise 5 on page 43 of Kunen

**definition** *indecomposable*

**where** *indecomposable*  $\alpha \equiv \text{Ord } \alpha \wedge (\forall \beta \in \text{elts } \alpha. \forall \gamma \in \text{elts } \alpha. \beta + \gamma \in \text{elts } \alpha)$

**lemma** *indecomposableD*:

$\llbracket \text{indecomposable } \alpha; \beta < \alpha; \gamma < \alpha; \text{Ord } \beta; \text{Ord } \gamma \rrbracket \implies \beta + \gamma < \alpha$   
*<proof>*

**lemma** *indecomposable-imp-Ord*:

*indecomposable*  $\alpha \implies \text{Ord } \alpha$   
*<proof>*

**lemma** *indecomposable-1*: *indecomposable 1*

*<proof>*

**lemma** *indecomposable-0*: *indecomposable 0*

*<proof>*

**lemma** *indecomposable-succ* [*simp*]: *indecomposable (succ  $\alpha$ )  $\longleftrightarrow \alpha = 0$*

*<proof>*

**lemma** *indecomposable-alt*:

**assumes** *ord*:  $\text{Ord } \alpha \text{ Ord } \beta$  **and**  $\beta: \beta < \alpha$  **and** *minor*:  $\bigwedge \beta \gamma. \llbracket \beta < \alpha; \gamma < \alpha; \text{Ord } \gamma \rrbracket \implies \beta + \gamma < \alpha$   
**shows**  $\beta + \alpha = \alpha$   
*<proof>*

**lemma** *indecomposable-imp-eq*:

**assumes** *indecomposable*  $\alpha \text{ Ord } \beta \beta < \alpha$   
**shows**  $\beta + \alpha = \alpha$   
*<proof>*

**lemma** *indecomposable2*:

**assumes**  $y: y < x$  **and**  $z: z < x$  **and** *minor*:  $\bigwedge y::V. y < x \implies y + x = x$   
**shows**  $y + z < x$   
*<proof>*

**lemma** *indecomposable-imp-Limit*:

**assumes** *indec*: *indecomposable*  $\alpha$  **and**  $\alpha > 1$   
**shows** *Limit*  $\alpha$   
*<proof>*

**lemma** *eq-imp-indecomposable*:

**assumes**  $\text{Ord } \alpha \bigwedge \beta::V. \beta \in \text{elts } \alpha \implies \beta + \alpha = \alpha$   
**shows** *indecomposable*  $\alpha$   
*<proof>*

**lemma** *indecomposable- $\omega$ -power*:

**assumes**  $\text{Ord } \delta$   
**shows** *indecomposable*  $(\omega \uparrow \delta)$   
*<proof>*

**lemma**  *$\omega$ -power-imp-eq:*

**assumes**  $\beta < \omega \uparrow \delta$  *Ord*  $\beta$  *Ord*  $\delta$   $\delta \neq 0$

**shows**  $\beta + \omega \uparrow \delta = \omega \uparrow \delta$

*<proof>*

**lemma** *mult-oexp-indec:*  $[[\text{Ord } \alpha; \text{Limit } \mu; \text{indecomposable } \mu]] \implies \alpha * (\alpha \uparrow \mu) = (\alpha \uparrow \mu)$

*<proof>*

**lemma** *mult-oexp- $\omega$ :* *Ord*  $\alpha \implies \alpha * (\alpha \uparrow \omega) = (\alpha \uparrow \omega)$

*<proof>*

**lemma** *type-imp-indecomposable:*

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

**and** *minor:*  $\bigwedge X. X \subseteq \text{elts } \alpha \implies \text{ordertype } X \text{ VWF} = \alpha \vee \text{ordertype } (\text{elts } \alpha - X) \text{ VWF} = \alpha$

**shows** *indecomposable*  $\alpha$

*<proof>*

This proof uses Cantor normal form, yet still is rather long

**proposition** *indecomposable-is- $\omega$ -power:*

**assumes** *inc:* *indecomposable*  $\mu$

**obtains**  $\mu = 0 \mid \delta$  **where** *Ord*  $\delta$   $\mu = \omega \uparrow \delta$

*<proof>*

**corollary** *indecomposable-iff- $\omega$ -power:*

*indecomposable*  $\mu \longleftrightarrow \mu = 0 \vee (\exists \delta. \mu = \omega \uparrow \delta \wedge \text{Ord } \delta)$

*<proof>*

**theorem** *indecomposable-imp-type:*

**fixes**  $X :: \text{bool} \Rightarrow V$  *set*

**assumes**  $\gamma$ : *indecomposable*  $\gamma$

**and**  $\bigwedge b. \text{ordertype } (X b) \text{ VWF} \leq \gamma \wedge b. \text{small } (X b) \wedge b. X b \subseteq ON$

**and**  $\text{elts } \gamma \subseteq (\bigcup b. X b)$

**shows**  $\exists b. \text{ordertype } (X b) \text{ VWF} = \gamma$

*<proof>*

**corollary** *indecomposable-imp-type2:*

**assumes**  $\alpha$ : *indecomposable*  $\gamma$   $X \subseteq \text{elts } \gamma$

**shows**  $\text{ordertype } X \text{ VWF} = \gamma \vee \text{ordertype } (\text{elts } \gamma - X) \text{ VWF} = \gamma$

*<proof>*

## 5.4 From ordinals to order types

**lemma** *indecomposable-ordertype-eq:*

**assumes** *indec:* *indecomposable*  $\alpha$  **and**  $\alpha$ : *ordertype*  $A$  *VWF* =  $\alpha$  **and**  $A: B \subseteq A$  *small*  $A$

**shows** *ordertype*  $B$  *VWF* =  $\alpha \vee \text{ordertype } (A - B) \text{ VWF} = \alpha$

*<proof>*

**lemma** *indecomposable-ordertype-ge*:

**assumes** *indec*: *indecomposable*  $\alpha$  **and**  $\alpha$ : *ordertype*  $A$  *VWF*  $\geq \alpha$  **and** *small*:  
*small*  $A$  *small*  $B$

**shows** *ordertype*  $B$  *VWF*  $\geq \alpha \vee$  *ordertype*  $(A-B)$  *VWF*  $\geq \alpha$   
*<proof>*

now for finite partitions

**lemma** *indecomposable-ordertype-finite-eq*:

**assumes** *indecomposable*  $\alpha$

**and**  $A$ : *finite*  $A$  *pairwise disjnt*  $A \cup A = A$   $A \neq \{\}$  *ordertype*  $A$  *VWF*  $= \alpha$   
*small*  $A$

**shows**  $\exists X \in A.$  *ordertype*  $X$  *VWF*  $= \alpha$   
*<proof>*

**lemma** *indecomposable-ordertype-finite-ge*:

**assumes** *indec*: *indecomposable*  $\alpha$

**and**  $A$ : *finite*  $A$   $A \subseteq \cup A$   $A \neq \{\}$  *ordertype*  $A$  *VWF*  $\geq \alpha$  *small*  $(\cup A)$

**shows**  $\exists X \in A.$  *ordertype*  $X$  *VWF*  $\geq \alpha$   
*<proof>*

**end**

## 6 Type Classes for ZFC

**theory** *ZFC-Typeclasses*

**imports** *ZFC-Cardinals* *Complex-Main*

**begin**

### 6.1 The class of embeddable types

**class** *embeddable* =

**assumes** *ex-inj*:  $\exists V\text{-of} :: 'a \Rightarrow V. \text{inj } V\text{-of}$

**context** *countable*

**begin**

**subclass** *embeddable*

*<proof>*

**end**

**instance** *unit* :: *embeddable* *<proof>*

**instance** *bool* :: *embeddable* *<proof>*

**instance** *nat* :: *embeddable* *<proof>*

**instance** *int* :: *embeddable* *<proof>*

**instance** *rat* :: *embeddable* *<proof>*

**instance** *char* :: *embeddable*  $\langle$ *proof* $\rangle$   
**instance** *String.literal* :: *embeddable*  $\langle$ *proof* $\rangle$   
**instance** *typerep* :: *embeddable*  $\langle$ *proof* $\rangle$

**lemma** *embeddable-classI*:  
**fixes**  $f :: 'a \Rightarrow V$   
**assumes**  $\bigwedge x y. f x = f y \implies x = y$   
**shows** *OFCLASS*('a, *embeddable-class*)  
 $\langle$ *proof* $\rangle$

**instance**  $V :: \textit{embeddable}$   
 $\langle$ *proof* $\rangle$

**instance** *prod* :: (*embeddable*,*embeddable*) *embeddable*  
 $\langle$ *proof* $\rangle$

**instance** *sum* :: (*embeddable*,*embeddable*) *embeddable*  
 $\langle$ *proof* $\rangle$

**instance** *option* :: (*embeddable*) *embeddable*  
 $\langle$ *proof* $\rangle$

**primrec** *V-of-list* **where**  
 $V\text{-of-list } V\text{-of Nil} = 0$   
 $| V\text{-of-list } V\text{-of } (x\#xs) = \langle V\text{-of } x, V\text{-of-list } V\text{-of } xs \rangle$

**lemma** *inj-V-of-list*:  
**assumes** *inj V-of*  
**shows** *inj (V-of-list V-of)*  
 $\langle$ *proof* $\rangle$

**instance** *list* :: (*embeddable*) *embeddable*  
 $\langle$ *proof* $\rangle$

## 6.2 The class of small types

**class** *small* =  
**assumes** *small*: *small* (*UNIV*::'a *set*)  
**begin**

**subclass** *embeddable*  
 $\langle$ *proof* $\rangle$

**lemma** *TC-small [iff]*:  
**fixes**  $A :: 'a \textit{ set}$   
**shows** *small A*  
 $\langle$ *proof* $\rangle$

```

end

context countable
begin

subclass small
  ⟨proof⟩

end

lemma lepoll-UNIV-imp-small:  $X \lesssim (UNIV::'a::small\ set) \implies small\ X$ 
  ⟨proof⟩

lemma lepoll-imp-small:
  fixes  $A :: 'a::small\ set$ 
  assumes  $X \lesssim A$ 
  shows small X
  ⟨proof⟩

instance unit :: small ⟨proof⟩
instance bool :: small ⟨proof⟩
instance nat :: small ⟨proof⟩
instance int :: small ⟨proof⟩
instance rat :: small ⟨proof⟩
instance char :: small ⟨proof⟩
instance String.literal :: small ⟨proof⟩
instance typerep :: small ⟨proof⟩

instance prod :: (small,small) small
  ⟨proof⟩

instance sum :: (small,small) small
  ⟨proof⟩

instance option :: (small) small
  ⟨proof⟩

instance list :: (small) small
  ⟨proof⟩

instance fun :: (small,embeddable) embeddable
  ⟨proof⟩

instance fun :: (small,small) small
  ⟨proof⟩

instance set :: (small) small
  ⟨proof⟩

```

**instance** *real* :: *small*  
<proof>

**instance** *complex* :: *small*  
<proof>

**end**

## 7 ZF sets corresponding to $\mathbb{R}$ and $\mathbb{C}$ and the cardinality of the continuum

**theory** *General-Cardinals*  
**imports** *ZFC-Typeclasses HOL-Analysis.Continuum-Not-Denumerable*

**begin**

### 7.1 Making the embedding from the type class explicit

**definition** *V-of* :: '*a*::*embeddable*  $\Rightarrow$  *V*  
**where** *V-of*  $\equiv$  *SOME* *f*. *inj* *f*

**lemma** *inj-V-of*: *inj* *V-of*  
<proof>

**declare** *inv-f-f* [*OF inj-V-of, simp*]

**lemma** *inv-V-of-image-eq* [*simp*]: *inv* *V-of* ' (*V-of* ' *X*) = *X*  
<proof>

**lemma** *infinite-V-of*: *infinite* (*UNIV*::'*a* *set*)  $\implies$  *infinite* (*range* (*V-of*::'*a*::*embeddable* $\Rightarrow$ *V*))  
<proof>

**lemma** *countable-V-of*: *countable* (*range* (*V-of*::'*a*::*countable* $\Rightarrow$ *V*))  
<proof>

**lemma** *elts-set-V-of*: *small* *X*  $\implies$  *elts* (*ZFC-in-HOL.set* (*V-of* ' *X*))  $\approx$  *X*  
<proof>

**lemma** *V-of-image-times*: *V-of* ' (*X*  $\times$  *Y*)  $\approx$  (*V-of* ' *X*)  $\times$  (*V-of* ' *Y*)  
<proof>

### 7.2 The cardinality of the continuum

**definition** *Real-set*  $\equiv$  *ZFC-in-HOL.set* (*range* (*V-of*::*real* $\Rightarrow$ *V*))

**definition** *Complex-set*  $\equiv$  *ZFC-in-HOL.set* (*range* (*V-of*::*complex* $\Rightarrow$ *V*))

**definition** *C-continuum*  $\equiv$  *vcard* *Real-set*

**lemma** *V-of-Real-set*: *bij-betw* *V-of* (*UNIV*::*real* *set*) (*elts* *Real-set*)  
<proof>

**lemma** *uncountable-Real-set: uncountable (elts Real-set)*  
*<proof>*

**lemma** *Card C-continuum*  
*<proof>*

**lemma** *C-continuum-ge: C-continuum  $\geq \aleph_1$*   
*<proof>*

**lemma** *V-of-Complex-set: bij-betw V-of (UNIV::complex set) (elts Complex-set)*  
*<proof>*

**lemma** *uncountable-Complex-set: uncountable (elts Complex-set)*  
*<proof>*

**lemma** *Complex-vcard: vcard Complex-set = C-continuum*  
*<proof>*

**lemma** *gcard-Union-le-cmult:*  
**assumes** *small U and  $\kappa: \bigwedge x. x \in U \implies \text{gcard } x \leq \kappa$  and sm:  $\bigwedge x. x \in U \implies$*   
*small x*  
**shows**  *$\text{gcard } (\bigcup U) \leq \text{gcard } U \otimes \kappa$*   
*<proof>*

**lemma** *gcard-Times [simp]:  $\text{gcard } (X \times Y) = \text{gcard } X \otimes \text{gcard } Y$*   
*<proof>*

### 7.3 Countable and uncountable sets

**lemma** *countable-iff-g-le-Aleph0:*  
**assumes** *small X*  
**shows**  *$\text{countable } X \longleftrightarrow \text{gcard } X \leq \aleph_0$*   
*<proof>*

**lemma** *countable-imp-g-le-Aleph0:  $\text{countable } X \implies \text{gcard } X \leq \aleph_0$*   
*<proof>*

**lemma** *finite-iff-g-le-Aleph0:  $\text{small } X \implies \text{finite } X \longleftrightarrow \text{gcard } X < \aleph_0$*   
*<proof>*

**lemma** *finite-imp-g-le-Aleph0:  $\text{finite } X \implies \text{gcard } X < \aleph_0$*   
*<proof>*

**lemma** *countable-infinite-gcard:  $\text{countable } X \wedge \text{infinite } X \longleftrightarrow \text{gcard } X = \aleph_0$*   
*<proof>*

**lemma** *uncountable-gcard:  $\text{small } X \implies \text{uncountable } X \longleftrightarrow \text{gcard } X > \aleph_0$*   
*<proof>*

**lemma** *uncountable-gcard-ge*:  $\text{small } X \implies \text{uncountable } X \longleftrightarrow \text{gcard } X \geq \aleph_1$   
*<proof>*

**lemma** *subset-smaller-gcard*:  
**assumes**  $\kappa: \kappa \leq \text{gcard } X \text{ Card } \kappa$   
**obtains**  $Y \text{ where } Y \subseteq X \text{ gcard } Y = \kappa$   
*<proof>*

**lemma** *Real-gcard*:  $\text{gcard } (\text{UNIV}::\text{real set}) = C\text{-continuum}$   
*<proof>*

**lemma** *Complex-gcard*:  $\text{gcard } (\text{UNIV}::\text{complex set}) = C\text{-continuum}$   
*<proof>*

**end**

## 8 Acknowledgements

The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

## References

- [1] L. Kirby. Addition and multiplication of sets. *Mathematical Logic Quarterly*, 53(1):52–65, 2007.
- [2] S. Obua. Partizan games in Isabelle/HOLZF. In K. Barkaoui, A. Cavalcanti, and A. Cerone, editors, *Theoretical Aspects of Computing — ICTAC 2006*, pages 272–286. Springer, 2006.