Zermelo Fraenkel Set Theory in Higher-Order Logic

Lawrence C. Paulson
Computer Laboratory
University of Cambridge

March 19, 2025

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

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-eqpoll-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-eqpoll-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?*]: $elts\ x = elts\ y \Longrightarrow x{=}y$
and *down-raw*: $Y \subseteq elts\ x \Longrightarrow Y \in range\ elts$
and *Union-raw*: $X \in range\ elts \Longrightarrow Union\ (elts\ `\ X) \in range\ elts$

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

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

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

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

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

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

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

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

Small classes can be mapped to sets.

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

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

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

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

lemma *Union* [*intro*]: $\mathit{small}\ X \Longrightarrow \mathit{small}\ (\mathit{Union}\ (\mathit{elts}\ `\ X))$
⟨*proof*⟩

lemma *Pow*: $\mathit{small}\ X \Longrightarrow \mathit{small}\ (\mathit{set}\ `\ \mathit{Pow}\ X)$
⟨*proof*⟩

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

lemma *replacement* [*intro*,*simp*]:
assumes *small X*
shows $\mathit{small}\ (f\ `\ X)$
⟨*proof*⟩

lemma *small-image-iff* [*simp*]: *inj-on f A* $\Longrightarrow$ *small* (*f* ‘ *A*) $\longleftrightarrow$ *small A*
⟨*proof*⟩

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

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

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

lemma *Finite-V*:
fixes *X* :: *V set*
assumes *finite X* **shows** *small X*
⟨*proof*⟩

lemma *small-insert-V*:
fixes *X* :: *V set*
assumes *small X*
shows *small* (*insert a X*)
⟨*proof*⟩

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

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

lemma *elts-vinsert* [*simp*]: *elts* (*vinsert x y*) = *insert x* (*elts y*)
⟨*proof*⟩

definition *succ* **where** *succ x* $\equiv$ *vinsert x x*

lemma *elts-succ* [*simp*]: *elts* (*succ x*) = *insert x* (*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 B* $\subseteq$ *A* **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* = *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* $\cup$ *elts y*)
⟨*proof*⟩

lemma *small-eqcong*: ⟦*small X*; *X* $\approx$ *Y*⟧ $\Longrightarrow$ *small Y*
⟨*proof*⟩

lemma *lepoll-small*: ⟦*small Y*; *X* $\lesssim$ *Y*⟧ $\Longrightarrow$ *small X*
⟨*proof*⟩

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

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

lemma *set-injective* [*simp*]: ⟦*small X*; *small Y*⟧ $\Longrightarrow$ *set X* = *set Y* $\longleftrightarrow$ *X*=*Y*
⟨*proof*⟩

1.2 Type classes and other basic setup

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

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

lemma *insert-neq-0* [*simp*]: *set* (*insert a X*) = *0* ⟷ ¬ *small X*
⟨*proof*⟩

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

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

definition *inf-V* **where** *inf-V x y* ≡ *set* (*elts x* ∩ *elts y*)

definition *sup-V* **where** *sup-V x y* ≡ *set* (*elts x* ∪ *elts y*)

definition *less-eq-V* **where** *less-eq-V x y* ≡ *elts x* ⊆ *elts y*

definition *less-V* **where** *less-V x y* ≡ *less-eq x y* ∧ *x* ≠ (*y*::*V*)

instance
⟨*proof*⟩
end

lemma *V-equalityI* [*intro*]: (⋀*x*. *x* ∈ *elts a* ⟷ *x* ∈ *elts b*) ⟹ *a* = *b*
⟨*proof*⟩

lemma *vsubsetI* [*intro*!]: (⋀*x*. *x* ∈ *elts a* ⟹ *x* ∈ *elts b*) ⟹ *a* ≤ *b*
⟨*proof*⟩

lemma *vsubsetD* [*elim*, *intro?*]: *a* ≤ *b* ⟹ *c* ∈ *elts a* ⟹ *c* ∈ *elts b*
⟨*proof*⟩

lemma *rev-vsubsetD*: *c* ∈ *elts a* ⟹ *a* ≤ *b* ⟹ *c* ∈ *elts b*
— The same, with reversed premises for use with *erule* – cf. ⟦*?P*; *?P* ⟶ *?Q*⟧ ⟹ *?Q*.
⟨*proof*⟩

lemma *vsubsetCE* [*elim*,*no-atp*]: *a* ≤ *b* ⟹ (*c* ∉ *elts a* ⟹ *P*) ⟹ (*c* ∈ *elts b* ⟹

$P) \Longrightarrow P$
— Classical elimination rule.
⟨*proof*⟩

lemma *set-image-le-iff*: $small\ A \Longrightarrow set\ (f\ `\ A) \leq B \longleftrightarrow (\forall x{\in}A.\ f\ x \in elts\ B)$
⟨*proof*⟩

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

lemma *less-eq-V-0-iff* [*simp*]: $x \leq 0 \longleftrightarrow x = 0$ **for** $x{::}V$
⟨*proof*⟩

lemma *subset-iff-less-eq-V*:
assumes $small\ B$ **shows** $A \subseteq B \longleftrightarrow set\ A \leq set\ B \land small\ A$
⟨*proof*⟩

lemma *small-Collect* [*simp*]: $small\ A \Longrightarrow small\ \{x \in A.\ P\ x\}$
⟨*proof*⟩

lemma *small-Union-iff*: $small\ (\bigcup (elts\ `\ X)) \longleftrightarrow small\ X$
⟨*proof*⟩

lemma *not-less-0* [*iff*]:
fixes $x{::}V$ **shows** $\neg\ x < 0$
⟨*proof*⟩

lemma *le-0* [*iff*]:
fixes $x{::}V$ **shows** $0 \leq x$
⟨*proof*⟩

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

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

lemma *neq0-conv*: $\bigwedge n{::}V.\ n \neq 0 \longleftrightarrow 0 < n$
⟨*proof*⟩

definition $VPow :: V \Rightarrow V$
where $VPow\ x \equiv set\ (set\ `\ Pow\ (elts\ x))$

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

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

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** $Sup\text{-}V\ X \equiv if\ small\ X\ then\ set\ (Union\ (elts\ `\ X))\ else\ 0$
instance ⟨*proof*⟩
end

instantiation V :: *Inf*
begin
definition *Inf-V* **where** $Inf\text{-}V\ X \equiv if\ X = \{\}\ then\ 0\ else\ set\ (Inter\ (elts\ `\ X))$
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*]: $bdd\text{-}above\ X = small\ X$ **for** $X::V\ set$
⟨*proof*⟩

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

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

instance
⟨*proof*⟩
end

lemma *Sup-upper*: $[\![x \in A;\ small\ A]\!] \Longrightarrow x \leq \bigsqcup A$ **for** $A::V\ set$
⟨*proof*⟩

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

⟨*proof*⟩

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

lemma *elts-Sup* [*simp*]: *small* $X \Longrightarrow$ *elts* $(\bigsqcup X) = \bigcup(elts \text{ ` } X)$
⟨*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** $\bigsqcup(insert\ x\ A) = x \sqcup \bigsqcup A$
⟨*proof*⟩

lemma *Sup-Un-distrib*: $[\![small\ A;\ small\ B]\!] \Longrightarrow \bigsqcup(A \cup B) = \bigsqcup A \sqcup \bigsqcup B$ **for** $A::V$ *set*
⟨*proof*⟩

lemma *SUP-sup-distrib*:
fixes $f :: V \Rightarrow V$
shows *small* $A \Longrightarrow (\bigsqcup x{\in}A.\ f\ x \sqcup g\ x) = \bigsqcup (f \text{ ` } A) \sqcup \bigsqcup (g \text{ ` } A)$
⟨*proof*⟩

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

lemma *cSUP-subset-mono*:
fixes $f :: 'a \Rightarrow V\ set$ **and** $g :: 'a \Rightarrow V\ set$
shows $[\![A \subseteq B;\ \bigwedge x.\ x \in A \Longrightarrow f\ x \le g\ x]\!] \Longrightarrow \bigsqcup (f \text{ ` } A) \le \bigsqcup (g \text{ ` } B)$
⟨*proof*⟩

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

lemma *cSUP-UNION*:
fixes $B :: V \Rightarrow V\ set$ **and** $f :: V \Rightarrow V$
assumes *ne*: *small A* **and** *bdd-UN*: *small* $(\bigcup x{\in}A.\ f \text{ ` } B\ x)$
shows $\bigsqcup(f \text{ ` } (\bigcup x{\in}A.\ B\ x)) = \bigsqcup((\lambda x.\ \bigsqcup(f \text{ ` } B\ x)) \text{ ` } A)$
⟨*proof*⟩

lemma *Sup-subset-mono*: *small* $B \Longrightarrow A \subseteq B \Longrightarrow Sup\ A \le Sup\ B$ **for** $A::V$ *set*
⟨*proof*⟩

lemma *Sup-le-iff*: *small* $A \Longrightarrow Sup\ A \le a \longleftrightarrow (\forall x{\in}A.\ x \le a)$ **for** $A::V$ *set*
⟨*proof*⟩

lemma *SUP-le-iff*: *small* $(f \text{ ` } A) \Longrightarrow \bigsqcup(f \text{ ` } A) \le u \longleftrightarrow (\forall x{\in}A.\ f\ x \le u)$ **for** f ::

$V \Rightarrow V$
  ⟨*proof*⟩

lemma *Sup-eq-0-iff* [*simp*]: $\bigsqcup A = 0 \longleftrightarrow A \subseteq \{0\} \vee \neg$ *small* A **for** $A :: V$ *set*
  ⟨*proof*⟩

lemma *Sup-Union-commute*:
  fixes $f :: V \Rightarrow V$ *set*
  assumes *small* A $\bigwedge x.\ x \in A \Longrightarrow$ *small* $(f\ x)$
  shows $\bigsqcup\ (\bigcup x \in A.\ f\ x) = (\bigsqcup x \in A.\ \bigsqcup\ (f\ x))$
  ⟨*proof*⟩

lemma *Sup-eq-Sup*:
  fixes $B :: V$ *set*
  assumes $B \subseteq A$ *small* A **and** $*$: $\bigwedge x.\ x \in A \Longrightarrow \exists\, y \in B.\ x \leq y$
  shows *Sup* A = *Sup* B
⟨*proof*⟩

1.3 Successor function

lemma *vinsert-not-empty* [*simp*]: *vinsert* $a\ A \neq 0$
  and *empty-not-vinsert* [*simp*]: $0 \neq$ *vinsert* $a\ A$
  ⟨*proof*⟩

lemma *succ-not-0* [*simp*]: *succ* $n \neq 0$ **and** *zero-not-succ* [*simp*]: $0 \neq$ *succ* n
  ⟨*proof*⟩

instantiation V :: *zero-neq-one*
begin
instance
  ⟨*proof*⟩
end

instantiation V :: *zero-less-one*
begin
instance
  ⟨*proof*⟩
end

lemma *succ-ne-self* [*simp*]: $i \neq$ *succ* i
  ⟨*proof*⟩

lemma *succ-notin-self*: *succ* $i \notin$ *elts* i
  ⟨*proof*⟩

lemma *le-succE*: *succ* $i \leq$ *succ* $j \Longrightarrow i \leq j$
  ⟨*proof*⟩

lemma *succ-inject-iff* [*iff*]: *succ* i = *succ* $j \longleftrightarrow i = j$

⟨*proof*⟩

lemma *inj-succ*: *inj succ*
⟨*proof*⟩

lemma *succ-neq-zero*: *succ* $x \neq 0$
⟨*proof*⟩

definition *pred* **where** *pred* $i \equiv$ *THE* $j.\ i =$ *succ* j

lemma *pred-succ* [*simp*]: *pred* (*succ* i) $= i$
⟨*proof*⟩

1.4 Ordinals

definition *Transset* **where** *Transset* $x \equiv \forall\, y \in$ *elts* $x.\ y \leq x$

definition *Ord* **where** *Ord* $x \equiv$ *Transset* $x \wedge (\forall\, y \in$ *elts* $x.$ *Transset* $y)$

abbreviation *ON* **where** *ON* $\equiv$ *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 $\bigwedge x.\ x \in X \Longrightarrow$ *Transset* x **shows** *Transset* $(\bigsqcup X)$
⟨*proof*⟩

lemma *Transset-sup*:
assumes *Transset* x *Transset* y **shows** *Transset* $(x \sqcup y)$
⟨*proof*⟩

lemma *Transset-inf*: $[\![$*Transset* i; *Transset* $j]\!] \Longrightarrow$ *Transset* $(i \sqcap j)$
⟨*proof*⟩

lemma *Transset-VPow*: *Transset*$(i) \Longrightarrow$ *Transset*(*VPow*(i))
⟨*proof*⟩

lemma *Transset-Inf*: $(\bigwedge i.\ i \in A \Longrightarrow$ *Transset* $i) \Longrightarrow$ *Transset* $(\bigsqcap A)$
⟨*proof*⟩

lemma *Transset-SUP*: $(\bigwedge x.\ x \in A \Longrightarrow$ *Transset* $(B\ x)) \Longrightarrow$ *Transset* $(\bigsqcup (B\ `\ A))$
⟨*proof*⟩

lemma *Transset-INT*: $(\bigwedge x.\ x \in A \Longrightarrow \mathit{Transset}\ (B\ x)) \Longrightarrow \mathit{Transset}\ (\bigsqcap\ (B\ `\ A))$
⟨*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 $\bigwedge x.\ x \in X \Longrightarrow \mathit{Ord}\ x$ **shows** $\mathit{Ord}\ (\bigsqcup X)$
⟨*proof*⟩

lemma *Ord-Union*:
assumes $\bigwedge x.\ x \in X \Longrightarrow \mathit{Ord}\ x$ *small X* **shows** $\mathit{Ord}\ (\mathit{set}\ (\bigcup\ (\mathit{elts}\ `\ X)))$
⟨*proof*⟩

lemma *Ord-sup*:
assumes *Ord x Ord y* **shows** $\mathit{Ord}\ (x \sqcup y)$
⟨*proof*⟩

lemma *big-ON* [*simp*]: $\neg$ *small ON*
⟨*proof*⟩

lemma *Ord-1* [*iff*]: *Ord 1*
⟨*proof*⟩

lemma *OrdmemD*: $\mathit{Ord}\ k \Longrightarrow j \in \mathit{elts}\ k \Longrightarrow j < k$
⟨*proof*⟩

lemma *Ord-trans*: $[\![\ i \in \mathit{elts}\ j;\ \ j \in \mathit{elts}\ k;\ \ \mathit{Ord}\ k\ ]\!] \Longrightarrow i \in \mathit{elts}\ k$
⟨*proof*⟩

lemma *mem-0-Ord*:
assumes *k*: *Ord k* **and** *knz*: $k \neq 0$ **shows** $0 \in \mathit{elts}\ k$
⟨*proof*⟩

lemma *Ord-in-Ord*: $[\![\ \mathit{Ord}\ k;\ \ m \in \mathit{elts}\ k\ ]\!] \Longrightarrow \mathit{Ord}\ m$
⟨*proof*⟩

lemma *OrdI*: $[\![\mathit{Transset}\ i;\ \bigwedge x.\ x \in \mathit{elts}\ i \Longrightarrow \mathit{Transset}\ x]\!] \Longrightarrow \mathit{Ord}\ i$
⟨*proof*⟩

lemma *Ord-is-Transset*: $\mathit{Ord}\ i \Longrightarrow \mathit{Transset}\ i$
⟨*proof*⟩

lemma *Ord-contains-Transset*: $[\![Ord\ i;\ j \in elts\ i]\!] \Longrightarrow Transset\ j$
⟨*proof*⟩

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

lemma *elts-subset-ON*: $Ord\ \alpha \Longrightarrow elts\ \alpha \subseteq ON$
⟨*proof*⟩

lemma *Transset-pred* [*simp*]: $Transset\ x \Longrightarrow \bigsqcup(elts\ (succ\ x)) = x$
⟨*proof*⟩

lemma *Ord-pred* [*simp*]: $Ord\ \beta \Longrightarrow \bigsqcup\ (insert\ \beta\ (elts\ \beta)) = \beta$
⟨*proof*⟩

1.4.3 Induction, Linearity, etc.

lemma *Ord-induct* [*consumes 1*, *case-names step*]:
assumes *k*: *Ord k*
and *step*: $\bigwedge x.[\![\ Ord\ x;\ \bigwedge y.\ y \in elts\ x \Longrightarrow P\ y\ ]\!] \ \Longrightarrow P\ x$
shows *P k*
⟨*proof*⟩

Comparability of ordinals

lemma *Ord-linear*: $Ord\ k \Longrightarrow Ord\ l \Longrightarrow k \in elts\ l \lor k{=}l \lor l \in elts\ k$
⟨*proof*⟩

The trichotomy law for ordinals

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

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

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

lemma *union-less-iff* [*simp*]: $[\![Ord\ i;\ Ord\ j]\!] \Longrightarrow i \sqcup j < k \longleftrightarrow i{<}k \land j{<}k$
⟨*proof*⟩

lemma *Ord-mem-iff-lt*: $Ord\ k \Longrightarrow Ord\ l \Longrightarrow k \in elts\ l \longleftrightarrow k < l$

⟨*proof*⟩

lemma *Ord-Collect-lt*: $Ord\ \alpha \Longrightarrow \{\xi.\ Ord\ \xi \wedge \xi < \alpha\} = elts\ \alpha$
⟨*proof*⟩

lemma *Ord-not-less*: $[\![Ord\ x;\ Ord\ y]\!] \Longrightarrow \neg\ x < y \longleftrightarrow y \le x$
⟨*proof*⟩

lemma *Ord-not-le*: $[\![Ord\ x;\ Ord\ y]\!] \Longrightarrow \neg\ x \le y \longleftrightarrow y < x$
⟨*proof*⟩

lemma *le-succ-iff*: $Ord\ i \Longrightarrow Ord\ j \Longrightarrow succ\ i \le succ\ j \longleftrightarrow i \le j$
⟨*proof*⟩

lemma *succ-le-iff*: $Ord\ i \Longrightarrow Ord\ j \Longrightarrow succ\ i \le j \longleftrightarrow i < j$
⟨*proof*⟩

lemma *succ-in-Sup-Ord*:
assumes *eq*: $succ\ \beta = \bigsqcup A$ **and** $small\ A\ A \subseteq ON\ Ord\ \beta$
shows $succ\ \beta \in A$
⟨*proof*⟩

lemma *in-succ-iff*: $Ord\ i \Longrightarrow j \in elts\ (ZFC\text{-}in\text{-}HOL.succ\ i) \longleftrightarrow Ord\ j \wedge j \le i$
⟨*proof*⟩

lemma *zero-in-succ* [*simp*,*intro*]: $Ord\ i \Longrightarrow 0 \in elts\ (succ\ i)$
⟨*proof*⟩

lemma *less-succ-self*: $x < succ\ x$
⟨*proof*⟩

lemma *Ord-finite-Sup*: $[\![finite\ A;\ A \subseteq ON;\ A \neq \{\}]\!] \Longrightarrow \bigsqcup A \in A$
⟨*proof*⟩

1.4.4 The natural numbers

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

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

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

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

lemma *Ord-equality*: $Ord\ i \Longrightarrow i = \bigsqcup\ (succ\ `\ elts\ i)$
⟨*proof*⟩

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

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

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

lemma *elts-ω*: $elts\ \omega = \{\alpha.\ \exists\, n.\ \alpha = ord\text{-}of\text{-}nat\ n\}$
⟨*proof*⟩

lemma *nat-into-Ord* [*simp*]: $n \in elts\ \omega \Longrightarrow Ord\ n$
⟨*proof*⟩

lemma *Sup-ω*: $\bigsqcup(elts\ \omega) = \omega$
⟨*proof*⟩

lemma *Ord-ω* [*iff*]: $Ord\ \omega$
⟨*proof*⟩

lemma *zero-in-omega* [*iff*]: $0 \in elts\ \omega$
⟨*proof*⟩

lemma *succ-in-omega* [*simp*]: $n \in elts\ \omega \Longrightarrow succ\ n \in elts\ \omega$
⟨*proof*⟩

lemma *ord-of-eq-0*: $ord\text{-}of\text{-}nat\ j = 0 \Longrightarrow j = 0$
⟨*proof*⟩

lemma *ord-of-nat-le-omega*: $ord\text{-}of\text{-}nat\ n \le \omega$
⟨*proof*⟩

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

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

corollary *inj-ord-of-nat*: $inj\ ord\text{-}of\text{-}nat$
⟨*proof*⟩

corollary *countable*:
assumes *countable X* **shows** *small X*
⟨*proof*⟩

corollary *infinite-ω*: *infinite* (*elts* ω)
⟨*proof*⟩

corollary *ord-of-nat-mono-iff* [*iff*]: *ord-of-nat* $i \leq$ *ord-of-nat* $j \longleftrightarrow i \leq j$
⟨*proof*⟩

corollary *ord-of-nat-strict-mono-iff* [*iff*]: *ord-of-nat* $i <$ *ord-of-nat* $j \longleftrightarrow i < j$
⟨*proof*⟩

lemma *small-image-nat* [*simp*]:
fixes N :: *nat set* **shows** *small* (g ‘ N)
⟨*proof*⟩

lemma *finite-Ord-omega*: $\alpha \in$ *elts* $\omega \Longrightarrow$ *finite* (*elts* α)
⟨*proof*⟩

lemma *infinite-Ord-omega*: *Ord* $\alpha \Longrightarrow$ *infinite* (*elts* α) $\Longrightarrow \omega \leq \alpha$
⟨*proof*⟩

lemma *ord-of-minus-1*: $n > 0 \Longrightarrow$ *ord-of-nat* $n =$ *succ* (*ord-of-nat* $(n - 1)$)
⟨*proof*⟩

lemma *card-ord-of-nat* [*simp*]: *card* (*elts* (*ord-of-nat* m)) $= m$
⟨*proof*⟩

lemma *ord-of-nat-ω* [*iff*]:*ord-of-nat* $n \in$ *elts* ω
⟨*proof*⟩

lemma *succ-ω-iff* [*iff*]: *succ* $n \in$ *elts* $\omega \longleftrightarrow n \in$ *elts* ω
⟨*proof*⟩

lemma *ω-gt0* [*simp*]: $\omega > 0$
⟨*proof*⟩

lemma *ω-gt1* [*simp*]: $\omega > 1$
⟨*proof*⟩

1.4.5 Limit ordinals

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

lemma *zero-not-Limit* [*iff*]: $\neg$ *Limit* *0*
⟨*proof*⟩

lemma *not-succ-Limit* [*simp*]: $\neg$ *Limit*(*succ* i)
⟨*proof*⟩

lemma *Limit-is-Ord*: *Limit* $\xi \Longrightarrow$ *Ord* ξ

⟨*proof*⟩

lemma *succ-in-Limit-iff*: $Limit\ \xi \Longrightarrow succ\ \alpha \in elts\ \xi \longleftrightarrow \alpha \in elts\ \xi$
⟨*proof*⟩

lemma *Limit-eq-Sup-self* [*simp*]: $Limit\ i \Longrightarrow Sup\ (elts\ i) = i$
⟨*proof*⟩

lemma *zero-less-Limit*: $Limit\ \beta \Longrightarrow 0 < \beta$
⟨*proof*⟩

lemma *non-Limit-ord-of-nat* [*iff*]: $\neg\ Limit\ (ord\text{-}of\text{-}nat\ m)$
⟨*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 $k = 0$ | l **where** $Ord\ l\ succ\ l = k$ | $Limit\ k$
⟨*proof*⟩

lemma *non-succ-LimitI*:
assumes $i{\neq}0$ $Ord(i)$ $\bigwedge y.\ succ(y) \neq i$
shows $Limit(i)$
⟨*proof*⟩

lemma *Ord-induct3* [*consumes 1*, *case-names 0 succ Limit*, *induct type*: V]:
assumes α: $Ord\ \alpha$
and P: $P\ 0$ $\bigwedge\alpha.\ [\![Ord\ \alpha;\ P\ \alpha]\!] \Longrightarrow P\ (succ\ \alpha)$
$\bigwedge\alpha.\ [\![Limit\ \alpha;\ \bigwedge\xi.\ \xi \in elts\ \alpha \Longrightarrow P\ \xi]\!] \Longrightarrow P\ (\bigsqcup\xi \in elts\ \alpha.\ \xi)$
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 $Ord\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
$\langle proof \rangle$
lemma *Ord-LeastI-ex*: $\exists\, i.\ Ord\ i \wedge P\ i \Longrightarrow P\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
  $\langle proof \rangle$

lemma *Ord-LeastI2*:
  $[\![Ord\ a;\ P\ a;\ \bigwedge x.\ [\![Ord\ x;\ P\ x]\!] \Longrightarrow Q\ x]\!] \Longrightarrow Q\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
  $\langle proof \rangle$

lemma *Ord-LeastI2-ex*:
  $\exists\, a.\ Ord\ a \wedge P\ a \Longrightarrow (\bigwedge x.\ [\![Ord\ x;\ P\ x]\!] \Longrightarrow Q\ x) \Longrightarrow Q\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
  $\langle proof \rangle$

lemma *Ord-LeastI2-wellorder*:
  assumes $Ord\ a\ P\ a$
  and $\bigwedge a.\ [\![\ P\ a;\ \forall\, b.\ Ord\ b \wedge P\ b \longrightarrow a \leq b\ ]\!] \Longrightarrow Q\ a$
  shows $Q\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
$\langle proof \rangle$

lemma *Ord-LeastI2-wellorder-ex*:
  assumes $\exists\, x.\ Ord\ x \wedge P\ x$
  and $\bigwedge a.\ [\![\ P\ a;\ \forall\, b.\ Ord\ b \wedge P\ b \longrightarrow a \leq b\ ]\!] \Longrightarrow Q\ a$
  shows $Q\ (LEAST\ i.\ Ord\ i \wedge P\ i)$
$\langle proof \rangle$

lemma *not-less-Ord-Least*: $[\![k < (LEAST\ x.\ Ord\ x \wedge P\ x);\ Ord\ k]\!] \Longrightarrow \neg\ P\ k$
  $\langle proof \rangle$

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** $?lhs \longleftrightarrow ?rhs$)
$\langle proof \rangle$

lemma *Ord-mono-imp-increasing*:
  assumes *fun-hD*: $h \in D \rightarrow D$
    and *mono-h*: $strict\text{-}mono\text{-}on\ D\ h$
    and $D \subseteq ON$ **and** ν: $\nu \in D$
  shows $\nu \leq h\ \nu$
$\langle proof \rangle$

lemma *le-Sup-iff*:
  assumes $A \subseteq ON\ Ord\ x\ small\ A$ **shows** $x \leq \bigsqcup A \longleftrightarrow (\forall\, y \in ON.\ y{<}x \longrightarrow (\exists\, a{\in}A.\ y < a))$
$\langle proof \rangle$

lemma *le-SUP-iff*: $[\![f\ `\ A \subseteq ON;\ Ord\ x;\ small\ A]\!] \Longrightarrow x \leq \bigsqcup (f\ `\ A) \longleftrightarrow (\forall\, y \in ON.\ y{<}x \longrightarrow (\exists\, i{\in}A.\ y < f\ i))$

⟨*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*:
$[\![\bigwedge x.\ f\ x \equiv transrec\ H\ x]\!] \Longrightarrow f\ a = H(\lambda x \in elts\ a.\ f\ x)\ a$
⟨*proof*⟩

lemma *eps-induct* [*case-names step*]:
assumes $\bigwedge x.\ (\bigwedge y.\ y \in elts\ x \Longrightarrow P\ y) \Longrightarrow P\ x$
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 $a \le b$ **shows** $Vfrom\ a\ i \le Vfrom\ b\ i$
⟨*proof*⟩

lemma *Vfrom-mono2*: $Vfrom\ a\ i \le Vfrom\ a\ (i \sqcup j)$
⟨*proof*⟩

lemma *Vfrom-mono*: $[\![Ord\ i;\ a{\le}b;\ i{\le}j]\!] \Longrightarrow Vfrom\ a\ i \le Vfrom\ b\ j$
⟨*proof*⟩

lemma *Transset-Vfrom*: $Transset(A) \Longrightarrow Transset(Vfrom\ A\ i)$
⟨*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*: $Ord\ i \Longrightarrow Vset(succ(i)) = VPow(Vset(i))$
⟨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*:
$Limit(i) \Longrightarrow Vfrom\ a\ i = (\bigsqcup y \in elts\ i.\ Vfrom\ a\ y)$
⟨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 :: $[V,\ Vs] \Rightarrow V$ (‹<(-,/ -)>›)
  -hpattern :: $[pttrn,\ patterns] \Rightarrow pttrn$ (‹<-,/ ->›)
syntax
  :: $V \Rightarrow Vs$ (‹-›)
  -Enum :: $[V,\ Vs] \Rightarrow Vs$ (‹-,/ -›)
  -Tuple :: $[V,\ Vs] \Rightarrow V$ (‹⟨(-,/ -)⟩›)
  -hpattern :: $[pttrn,\ patterns] \Rightarrow pttrn$ (‹⟨-,/ -⟩›)
syntax-consts
  -Enum -Tuple ⇌ *vpair* **and**
  -hpattern ⇌ *vsplit*
translations
  $<x,\ y,\ z> \ \rightleftharpoons\ <x,\ <y,\ z>>$
  $<x,\ y> \ \rightleftharpoons\ CONST\ vpair\ x\ y$
  $<x,\ y,\ z> \ \rightleftharpoons\ <x,\ <y,\ z>>$
  $\lambda<x,y,zs>.\ b \rightleftharpoons CONST\ vsplit(\lambda x\ <y,zs>.\ b)$
  $\lambda<x,y>.\ b \rightleftharpoons CONST\ vsplit(\lambda x\ y.\ b)$

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

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

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

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

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

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

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

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

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

lemma *zero-notin-vpair*: $0 \notin elts\ \langle x,y\rangle$
 $\langle proof\rangle$

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

2.2 Generalized Cartesian product

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

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

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

lemma *pairs-iff-elts*: $(x,y) \in pairs\ z \longleftrightarrow \langle x,y\rangle \in elts\ z$
 $\langle proof\rangle$

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

lemma *VSigmaI* [*intro*!]: $[\![\ a \in elts\ A;\ \ b \in elts\ (B\ a)]\!] \Longrightarrow \langle a,b\rangle \in elts\ (VSigma\ A\ B)$
 $\langle proof\rangle$

lemmas *VSigmaD1* = *VSigma-iff* [*THEN iffD1*, *THEN conjunct1*]
lemmas *VSigmaD2* = *VSigma-iff* [*THEN iffD1*, *THEN conjunct2*]

The general elimination rule

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

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

lemma *VSigma-empty1* [*simp*]: $VSigma\ 0\ B = 0$
 $\langle proof\rangle$

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

lemma *timesI* [*intro*!]: $[\![a \in elts\ A;\ \ b \in elts\ B]\!] \Longrightarrow \langle a,b\rangle \in elts\ (vtimes\ A\ B)$
⟨*proof*⟩

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

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

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

lemma *small-Sigma* [*simp*]:
assumes *A*: *small A* **and** *B*: $\bigwedge x.\ x \in A \Longrightarrow small\ (B\ x)$
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*: $small\ (A \times B) \longleftrightarrow small\ A \wedge small\ B \vee A{=}\{\} \vee B{=}\{\}$
(**is** - = *?rhs*)
⟨*proof*⟩

2.3 Disjoint Sum

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

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

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

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

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

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

2.3.1 Equivalences for the injections and an elimination rule

lemma *Inl-in-sum-iff* [*iff*]: $Inl\ a \in elts\ (A \uplus B) \longleftrightarrow a \in elts\ A$
⟨*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** $x \in elts\ A\ u{=}Inl\ x$ | 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*: $A \uplus B \le C \uplus D \longleftrightarrow A{\le}C \wedge B{\le}D$
⟨*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 $is\text{-}sum\ z = (\exists x.\ z = Inl\ x \vee z = Inr\ x)$

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

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

lemma *sum-case-Inl* [*simp*]: $sum\text{-}case\ f\ g\ (Inl\ x) = f\ x$
⟨*proof*⟩

lemma *sum-case-Inr* [*simp*]: $sum\text{-}case\ f\ g\ (Inr\ y) = g\ y$
⟨*proof*⟩

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

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

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

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

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

2.4 Generalised function space and lambda

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

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

where *app f x* ≡ *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* ≡ *set* {$f \in$ *elts* (*VPow*(*VSigma A B*)). *elts A* ≤ *Domain* (*pairs f*) ∧ *single-valued* (*pairs f*)}

lemma *VPi-I*:
assumes $\bigwedge x.\ x \in$ *elts A* ⟹ $b\ x \in$ *elts* ($B\ x$)
shows *VLambda A b* ∈ *elts* (*VPi A B*)
⟨*proof*⟩

lemma *apply-pair*:
assumes *f*: $f \in$ *elts* (*VPi A B*) **and** *x*: $x \in$ *elts A*
shows ⟨*x*, *app f x*⟩ ∈ *elts f*
⟨*proof*⟩

lemma *VPi-D*:
assumes *f*: $f \in$ *elts* (*VPi A B*) **and** *x*: $x \in$ *elts A*
shows *app f x* ∈ *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* $p =$ ⟨*x*, *app f x*⟩
⟨*proof*⟩

lemma *fun-ext*:
assumes $f \in$ *elts* (*VPi A B*) $g \in$ *elts* (*VPi A B*) $\bigwedge x.\ x \in$ *elts A* ⟹ *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*: $VLambda\ A\ f = VLambda\ B\ g \Longrightarrow A = B$
⟨*proof*⟩

lemma *VLambda-eq-D2*: $[\![VLambda\ A\ f = VLambda\ A\ g;\ x \in elts\ A]\!] \Longrightarrow f\ x = g\ x$
⟨*proof*⟩

2.5 Transitive closure of a set

definition $TC :: V \Rightarrow V$
where $TC \equiv transrec\ (\lambda f\ x.\ x \sqcup \bigsqcup\ (f\ `\ elts\ x))$

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

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

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

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

lemma *TC-least*: $[\![Transset\ x;\ \ a \le x]\!] \Longrightarrow TC\ a \le x$
⟨*proof*⟩

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

definition *le-TC* (**infix** ‹$\sqsubseteq$› *50*)
where $x \sqsubseteq y \equiv x \sqsubset y \lor x = y$

lemma *less-TC-imp-not-le*: $x \sqsubset a \Longrightarrow \neg\ a \le x$
⟨*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*: $x \neq 0 \Longrightarrow 0 \sqsubset x$
⟨*proof*⟩

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

lemma *less-asym-TC*: $[\![x \sqsubset y;\ y \sqsubset x]\!] \Longrightarrow False$

⟨*proof*⟩

lemma *le-antisym-TC*: $[\![x \sqsubseteq y;\ y \sqsubseteq x]\!] \Longrightarrow x = y$
⟨*proof*⟩

lemma *less-le-TC*: $x \sqsubset y \longleftrightarrow x \sqsubseteq y \wedge x \neq y$
⟨*proof*⟩

lemma *less-imp-le-TC* [*iff*]: $x \sqsubset y \Longrightarrow x \sqsubseteq y$
⟨*proof*⟩

lemma *le-TC-refl* [*iff*]: $x \sqsubseteq x$
⟨*proof*⟩

lemma *le-TC-trans* [*trans*]: $[\![x \sqsubseteq y;\ y \sqsubseteq z]\!] \Longrightarrow x \sqsubseteq z$
⟨*proof*⟩

context *order*
begin

lemma *nless-le-TC*: $(\neg\ a \sqsubset b) \longleftrightarrow (\neg\ a \sqsubseteq b) \vee a = b$
⟨*proof*⟩

lemma *eq-refl-TC*: $x = y \Longrightarrow x \sqsubseteq y$
⟨*proof*⟩

⟨*ML*⟩

end

lemma *less-TC-trans* [*trans*]: $[\![x \sqsubset y;\ y \sqsubset z]\!] \Longrightarrow x \sqsubset z$
and *less-le-TC-trans*: $[\![x \sqsubset y;\ y \sqsubseteq z]\!] \Longrightarrow x \sqsubset z$
and *le-less-TC-trans* [*trans*]: $[\![x \sqsubseteq y;\ y \sqsubset z]\!] \Longrightarrow x \sqsubset z$
⟨*proof*⟩

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

lemma *TC-Sup-distrib*:
assumes *small X* **shows** $TC\ (\bigsqcup X) = \bigsqcup (TC\ `\ X)$
⟨*proof*⟩

lemma *TC'*: $TC\ x = x \sqcup TC\ (\bigsqcup\ (elts\ x))$
⟨*proof*⟩

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

A distinctive induction principle

lemma *TC-induct-down-lemma*:
assumes *ab*: $a \sqsubset b$ **and** *base*: $b \le d$
and *step*: $\bigwedge y\ z.\ [\![y \sqsubset b;\ y \in elts\ d;\ z \in elts\ y]\!] \Longrightarrow z \in elts\ d$
shows $a \in elts\ d$
⟨*proof*⟩

lemma *TC-induct-down* [*consumes 1*, *case-names base step small*]:
assumes $a \sqsubset b$
and $\bigwedge y.\ y \in elts\ b \Longrightarrow P\ y$
and $\bigwedge y\ z.\ [\![y \sqsubset b;\ P\ y;\ z \in elts\ y]\!] \Longrightarrow P\ z$
and *small* (*Collect P*)
shows $P\ a$
⟨*proof*⟩

2.6 Rank of a set

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

lemma *rank*: $rank\ a = set(\bigcup y \in elts\ a.\ elts\ (succ(rank\ y)))$
⟨*proof*⟩

lemma *rank-Sup*: $rank\ a = \bigsqcup((\lambda y.\ succ(rank\ y))\ `\ elts\ a)$
⟨*proof*⟩

lemma *Ord-rank* [*simp*]: $Ord(rank\ a)$
⟨*proof*⟩

lemma *rank-of-Ord*: $Ord\ i \Longrightarrow rank\ i = i$
⟨*proof*⟩

lemma *Ord-iff-rank*: $Ord\ x \longleftrightarrow rank\ x = x$
⟨*proof*⟩

lemma *rank-lt*: $a \in elts\ b \Longrightarrow rank\ a < rank\ b$
⟨*proof*⟩

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

lemma *rank-succ* [*simp*]: $rank(succ\ x) = succ(rank\ x)$
⟨*proof*⟩

lemma *rank-mono*: $a \le b \Longrightarrow rank\ a \le rank\ b$
⟨*proof*⟩

lemma *VsetI*: $rank\ b \sqsubset i \Longrightarrow b \in elts\ (Vset\ i)$
⟨*proof*⟩

lemma *Ord-VsetI*: $[\![Ord\ i;\ rank\ b < i]\!] \Longrightarrow b \in elts\ (Vset\ i)$
⟨*proof*⟩

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

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

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

lemma *small-ranks-imp-small*:
assumes *small* (*rank* ‘ *A*) **shows** *small A*
⟨*proof*⟩

lemma *rank-Union*: $rank(\bigsqcup\ A) = \bigsqcup\ (rank\ `\ A)$
⟨*proof*⟩

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

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

lemma *TC-rank-lt*: $a \sqsubset b \Longrightarrow rank\ a < rank\ b$
⟨*proof*⟩

lemma *TC-rank-mem*: $x \sqsubset y \Longrightarrow rank\ x \in elts\ (rank\ y)$
⟨*proof*⟩

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

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

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

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

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

⟨*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-VWO*: $wf(VWO - Id)$
⟨*proof*⟩

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

lemma *refl-VWO*: *refl VWO*
⟨*proof*⟩

lemma *trans-VWO*: *trans VWO*
⟨*proof*⟩

lemma *antisym-VWO*: *antisym VWO*
⟨*proof*⟩

lemma *total-VWO*: *total VWO*
⟨*proof*⟩

lemma *total-VWOId*: *total* ($VWO-Id$)
⟨*proof*⟩

lemma *Linear-order-VWO*: *Linear-order VWO*
⟨*proof*⟩

lemma *wo-rel-VWO*: *wo-rel VWO*
⟨*proof*⟩

2.7.1 Transitive Closure and VWO

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

lemma *less-TC-imp-VWO*: $x \sqsubset y \Longrightarrow (x,y) \in VWO$
⟨*proof*⟩

lemma *le-TC-imp-VWO*: $x \sqsubseteq y \Longrightarrow (x,y) \in VWO$
⟨*proof*⟩

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

lemma *less-TC-succ*: $x \sqsubset succ\ \beta \longleftrightarrow x \sqsubset \beta \lor x = \beta$
⟨*proof*⟩

lemma *le-TC-succ*: $x \sqsubseteq succ\ \beta \longleftrightarrow x \sqsubseteq \beta \lor x = succ\ \beta$
⟨*proof*⟩

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

lemma *Ord-TC-less-iff*: $[\![Ord\ \alpha;\ Ord\ \beta]\!] \Longrightarrow \beta \sqsubset \alpha \longleftrightarrow \beta < \alpha$
⟨*proof*⟩

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

lemma *le-TC-Ord*: $[\![\beta \sqsubseteq \alpha;\ Ord\ \alpha]\!] \Longrightarrow Ord\ \beta$
⟨*proof*⟩

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

lemma *VWO-TC-le*: $[\![Ord\ \alpha;\ Ord\ \beta;\ (\beta,\ \alpha) \in VWO]\!] \Longrightarrow \beta \sqsubseteq \alpha$
⟨*proof*⟩

lemma *VWO-iff-Ord-le* [*simp*]: $[\![Ord\ \alpha;\ Ord\ \beta]\!] \Longrightarrow (\beta,\ \alpha) \in VWO \longleftrightarrow \beta \le \alpha$
⟨*proof*⟩

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

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

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

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

2.7.2 Relation VWF

definition *VWF* **where** $VWF \equiv VWO - Id$

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*]: $[\![Ord\ \alpha;\ Ord\ \beta]\!] \Longrightarrow (\alpha,\beta) \in VWF \longleftrightarrow \alpha < \beta$
⟨*proof*⟩

lemma *mem-imp-VWF*: $x \in elts\ y \Longrightarrow (x,y) \in VWF$
⟨*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*:
$small\ A \Longrightarrow ordermap\ A\ r \in A \rightarrow elts\ (ordertype\ A\ r)$
⟨*proof*⟩

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

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

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

x)
⟨*proof*⟩

lemma *wf-Ord-ordertype*: **assumes** *wf r trans r* **shows** *Ord*(*ordertype A r*)
⟨*proof*⟩

lemma *Ord-ordertype* [*simp*]: *Ord*(*ordertype A VWF*)
⟨*proof*⟩

lemma *Ord-ordermap* [*simp*]: *Ord* (*ordermap A VWF x*)
⟨*proof*⟩

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

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*)
⟨*proof*⟩

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 $(y, x) \in r$
⟨*proof*⟩

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*) $\longleftrightarrow$ $(y, x) \in r$
⟨*proof*⟩

lemma *ordermap-surj*: *elts* (*ordertype A r*) $\subseteq$ *ordermap A r* ‘ *A*
⟨*proof*⟩

lemma *ordermap-bij*:
assumes *wf r total-on A r small A*
shows *bij-betw* (*ordermap A r*) *A* (*elts* (*ordertype A r*))
⟨*proof*⟩

lemma *ordermap-eq-iff* [*simp*]:
⟦$x \in A$; $y \in A$; *wf r*; *total-on A r*; *small A*⟧ $\Longrightarrow$ *ordermap A r x* = *ordermap A r y* $\longleftrightarrow$ $x = y$
⟨*proof*⟩

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

$r)$ $\alpha \in A$
⟨*proof*⟩

lemma *ordertype-nat-imp-finite*:
assumes *ordertype A r = ord-of-nat m small A wf r total-on A r*
shows *finite A*
⟨*proof*⟩

lemma *wf-ordertype-eqpoll*:
assumes *wf r total-on A r small A*
shows *elts (ordertype A r) ≈ A*
⟨*proof*⟩

lemma *ordertype-eqpoll*:
assumes *small A*
shows *elts (ordertype A VWF) ≈ A*
⟨*proof*⟩

2.9 More advanced *ordertype* and *ordermap* results

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

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

lemma *ordertype-eq-0-iff* [*simp*]: ⟦*small X*; *wf r*⟧ ⟹ *ordertype X r = 0* ⟷ *X = {}*
⟨*proof*⟩

lemma *ordermap-mono-less*:
assumes *(w, x) ∈ r*
and *wf r trans r*
and *w ∈ A x ∈ A*
and *small A*
shows *ordermap A r w < ordermap A r x*
⟨*proof*⟩

lemma *ordermap-mono-le*:
assumes *(w, x) ∈ r ∨ w=x*
and *wf r trans r*
and *w ∈ A x ∈ A*
and *small A*
shows *ordermap A r w ≤ ordermap A r x*
⟨*proof*⟩

lemma *converse-ordermap-le-mono*:
assumes *ordermap A r y ≤ ordermap A r x wf r total r x ∈ A small A*
shows *(y, x) ∈ r ∨ y=x*

$\langle proof \rangle$

lemma *ordertype-mono*:
assumes $X \subseteq Y$ **and** r: *wf r trans r* **and** *small Y*
shows *ordertype X r* $\leq$ *ordertype Y r*
$\langle proof \rangle$

corollary *ordertype-VWF-mono*:
assumes $X \subseteq Y$ *small Y*
shows *ordertype X VWF* $\leq$ *ordertype Y VWF*
$\langle proof \rangle$

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*
$\langle proof \rangle$

lemma *inv-ordermap-mono-less*:
assumes (*inv-into M* (*ordermap M r*) α, *inv-into M* (*ordermap M r*) β) $\in r$
and *small M* **and** α: $\alpha \in$ *elts* (*ordertype M r*) **and** β: $\beta \in$ *elts* (*ordertype M r*)
and *wf r trans r*
shows $\alpha < \beta$
$\langle proof \rangle$

lemma *inv-ordermap-mono-eq*:
assumes *inv-into M* (*ordermap M r*) α = *inv-into M* (*ordermap M r*) β
and $\alpha \in$ *elts* (*ordertype M r*) $\beta \in$ *elts* (*ordertype M r*)
shows $\alpha = \beta$
$\langle proof \rangle$

lemma *inv-ordermap-VWF-mono-le*:
assumes *inv-into M* (*ordermap M VWF*) $\alpha \leq$ *inv-into M* (*ordermap M VWF*) β
and $M \subseteq$ *ON small M* **and** α: $\alpha \in$ *elts* (*ordertype M VWF*) **and** β: $\beta \in$ *elts* (*ordertype M VWF*)
shows $\alpha \leq \beta$
$\langle proof \rangle$

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*) $\beta \longleftrightarrow \alpha \leq \beta$
$\langle proof \rangle$

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*) β

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

lemma *strict-mono-on-ordertype*:
assumes $M \subseteq ON$ *small* M
obtains f **where** $f \in$ *elts* (*ordertype* M *VWF*) $\to M$ *strict-mono-on* (*elts* (*ordertype* M *VWF*)) f
⟨*proof*⟩

lemma *ordermap-inc-eq*:
assumes $x \in A$ *small* A
and π: $\bigwedge x\ y.\ [\![x{\in}A;\ y{\in}A;\ (x,y) \in r]\!] \Longrightarrow (\pi\ x,\ \pi\ y) \in s$
and r: *wf* r *total-on* A r **and** *wf* s
shows *ordermap* (π ‘ A) s (π x) = *ordermap* A r x
⟨*proof*⟩

lemma *ordertype-inc-eq*:
assumes *small* A
and π: $\bigwedge x\ y.\ [\![x{\in}A;\ y{\in}A;\ (x,y) \in r]\!] \Longrightarrow (\pi\ x,\ \pi\ y) \in s$
and r: *wf* r *total-on* A r **and** *wf* s
shows *ordertype* (π ‘ A) s = *ordertype* A r
⟨*proof*⟩

lemma *ordertype-inc-le*:
assumes *small* A *small* B
and π: $\bigwedge x\ y.\ [\![x{\in}A;\ y{\in}A;\ (x,y) \in r]\!] \Longrightarrow (\pi\ x,\ \pi\ y) \in s$
and r: *wf* r *total-on* A r **and** *wf* s *trans* s
and π ‘ $A \subseteq B$
shows *ordertype* A $r \le$ *ordertype* B s
⟨*proof*⟩

corollary *ordertype-VWF-inc-eq*:
assumes $A \subseteq ON$ π ‘ $A \subseteq ON$ *small* A **and** $\bigwedge x\ y.\ [\![x{\in}A;\ y{\in}A;\ x{<}y]\!] \Longrightarrow \pi\ x < \pi\ y$
shows *ordertype* (π ‘ A) *VWF* = *ordertype* A *VWF*
⟨*proof*⟩

lemma *ordertype-image-ordermap*:
assumes *small* A $X \subseteq A$ *wf* r *trans* r *total-on* X r
shows *ordertype* (*ordermap* A r ‘ X) *VWF* = *ordertype* X r
⟨*proof*⟩

lemma *ordertype-map-image*:
assumes $B \subseteq A$ *small* A
shows *ordertype* (*ordermap* A *VWF* ‘ A − *ordermap* A *VWF* ‘ B) *VWF* = *ordertype* ($A - B$) *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* $\longleftrightarrow$
$(\exists f \in A \to B.\ \mathit{inj\text{-}on}\ f\ A \wedge (\forall x \in A.\ \forall y \in A.\ ((x,y) \in r \longrightarrow (f\ x,\ f\ y) \in s)))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

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′*
⟨*proof*⟩

lemma *ordertype-infinite-ge-*ω:
assumes *infinite A small A*
shows *ordertype A VWF* $\geq \omega$
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *ordermap-eq-self*:
assumes *Ord* α **and** x: $x \in$ *elts* α
shows *ordermap* (*elts* α) *VWF* $x = x$
⟨*proof*⟩

lemma *ordertype-eq-Ord* [*simp*]:
assumes *Ord* α
shows *ordertype* (*elts* α) *VWF* = α
⟨*proof*⟩

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

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

$\langle proof \rangle$

lemma *ordertype-le-Ord*:
assumes *Ord* α $X \subseteq$ *elts* α
shows *ordertype* X *VWF* $\le \alpha$
$\langle proof \rangle$

lemma *ordertype-inc-le-Ord*:
assumes *small* A *Ord* α
and π: $\bigwedge x\ y.\ [\![x{\in}A;\ y{\in}A;\ (x,y) \in r]\!] \Longrightarrow \pi\ x < \pi\ y$
and *wf r total-on A r*
and *sub*: π ' $A \subseteq$ *elts* α
shows *ordertype* A $r \le \alpha$
$\langle proof \rangle$

lemma *le-ordertype-obtains-subset*:
assumes α: $\beta \le \alpha$ *ordertype* H *VWF* $= \alpha$ **and** *small* H *Ord* β
obtains G **where** $G \subseteq H$ *ordertype* G *VWF* $= \beta$
$\langle proof \rangle$

lemma *ordertype-infinite-* ω:
assumes $A \subseteq$ *elts* ω *infinite* A
shows *ordertype* A *VWF* $= \omega$
$\langle proof \rangle$

For infinite sets of natural numbers

lemma *ordertype-nat-* ω:
assumes *infinite* N **shows** *ordertype* N *less-than* $= \omega$
$\langle proof \rangle$

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*)
$\langle proof \rangle$

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*)
$\langle proof \rangle$

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) $= A$

$\langle proof \rangle$

lemma *ordertype-Field-Restr*:
assumes $wf\ r\ total\text{-}on\ A\ r\ trans\ r\ small\ A\ Field\ (Restr\ r\ A) = A$
shows $ordertype\ (Field\ (Restr\ r\ A))\ (Restr\ r\ A) = ordertype\ A\ r$
$\langle proof \rangle$

proposition *ordertype-eq-ordertype-iso-Restr*:
assumes r: $wf\ r\ total\text{-}on\ A\ r\ trans\ r$ **and** $small\ A$ **and** FA: $Field\ (Restr\ r\ A) = A$
assumes s: $wf\ s\ total\text{-}on\ B\ s\ trans\ s$ **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** $?lhs = ?rhs$)
$\langle proof \rangle$

lemma *ordermap-insert*:
assumes $Ord\ \alpha$ **and** y: $Ord\ y\ y \le \alpha$ **and** U: $U \subseteq elts\ \alpha$
shows $ordermap\ (insert\ \alpha\ U)\ VWF\ y = ordermap\ U\ VWF\ y$
$\langle proof \rangle$

lemma *ordertype-insert*:
assumes $Ord\ \alpha$ **and** U: $U \subseteq elts\ \alpha$
shows $ordertype\ (insert\ \alpha\ U)\ VWF = succ\ (ordertype\ U\ VWF)$
$\langle proof \rangle$

lemma *finite-ordertype-le-card*:
assumes $finite\ A\ wf\ r\ trans\ r$
shows $ordertype\ A\ r \le ord\text{-}of\text{-}nat\ (card\ A)$
$\langle proof \rangle$

lemma *ordertype-VWF-ω*:
assumes $finite\ A$
shows $ordertype\ A\ VWF \in elts\ \omega$
$\langle proof \rangle$

lemma *ordertype-VWF-finite-nat*:
assumes $finite\ A$
shows $ordertype\ A\ VWF = ord\text{-}of\text{-}nat\ (card\ A)$
$\langle proof \rangle$

lemma *finite-ordertype-eq-card*:
assumes $small\ A\ wf\ r\ trans\ r\ total\text{-}on\ A\ r$
shows $ordertype\ A\ r = ord\text{-}of\text{-}nat\ m \longleftrightarrow finite\ A \land card\ A = m$
$\langle proof \rangle$

lemma *ex-bij-betw-strict-mono-card*:
assumes $finite\ M\ M \subseteq ON$

obtains h **where** $\textit{bij-betw } h \ \{..<\textit{card } M\} \ M$ **and** $\textit{strict-mono-on } \{..<\textit{card } M\} \ h$
⟨*proof*⟩

lemma *ordertype-finite-less-than* [*simp*]:
assumes *finite A* **shows** $\textit{ordertype } A \textit{ less-than} = \textit{card } A$
⟨*proof*⟩

2.10 Cardinality of an arbitrary HOL set

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

2.11 Cardinality of a set

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

lemma *gcard-eq-vcard* [*simp*]: $\textit{gcard } (\textit{elts } x) = \textit{vcard } x$
⟨*proof*⟩

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

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

lemma *cardinal-cong*: $\textit{elts } x \approx \textit{elts } y \Longrightarrow \textit{vcard } x = \textit{vcard } y$
⟨*proof*⟩

lemma *gcardinal-cong*:
assumes $X \approx Y$ **shows** $\textit{gcard } X = \textit{gcard } Y$
⟨*proof*⟩

lemma *vcard-set-image*: $\textit{inj-on } f \ (\textit{elts } x) \Longrightarrow \textit{vcard } (\textit{set } (f \text{ ` } \textit{elts } x)) = \textit{vcard } x$
⟨*proof*⟩

lemma *gcard-image*: $\textit{inj-on } f \ X \Longrightarrow \textit{gcard } (f \text{ ` } X) = \textit{gcard } X$
⟨*proof*⟩

lemma *Card-cardinal-eq*: $\textit{Card } \kappa \Longrightarrow \textit{vcard } \kappa = \kappa$
⟨*proof*⟩

lemma *Card-is-Ord*:
assumes $\textit{Card } \kappa$ **shows** $\textit{Ord } \kappa$
⟨*proof*⟩

lemma *cardinal-eqpoll*: $\textit{elts } (\textit{vcard } a) \approx \textit{elts } a$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *cardinal-idem* [*simp*]: $vcard\ (vcard\ a) = vcard\ a$
⟨*proof*⟩

lemma *subset-smaller-vcard*:
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*]: $Card\ (ord\text{-}of\text{-}nat\ n)$
⟨*proof*⟩

lemma *Card-0* [*iff*]: $Card\ 0$
⟨*proof*⟩

lemma *CardI*: $[\![Ord\ i;\ \bigwedge j.\ [\![j < i;\ Ord\ j]\!] \Longrightarrow \neg\ elts\ j \approx elts\ i]\!] \Longrightarrow Card\ i$
⟨*proof*⟩

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

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

lemma *gcard-big-0*: $\neg\ small\ X \Longrightarrow gcard\ X = 0$
⟨*proof*⟩

lemma *gcard-eq-card*:
assumes $finite\ X$ **shows** $gcard\ X = ord\text{-}of\text{-}nat\ (card\ X)$
⟨*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-eqpoll*: $small\ X \Longrightarrow elts\ (gcard\ X) \approx X$
⟨*proof*⟩

lemma *lepoll-imp-gcard-le*:
  assumes $A \lesssim B$ *small B*
  shows *gcard A* $\leq$ *gcard B*
⟨*proof*⟩

lemma *gcard-image-le*:
  assumes *small A* **shows** *gcard* (f ‘ A) $\leq$ *gcard A*
  ⟨*proof*⟩

lemma *subset-imp-gcard-le*:
  assumes $A \subseteq B$ *small B*
  shows *gcard A* $\leq$ *gcard B*
  ⟨*proof*⟩

lemma *gcard-le-lepoll*: $[\![$*gcard A* $\leq \alpha$; *small A*$]\!] \Longrightarrow A \lesssim$ *elts* α
  ⟨*proof*⟩

2.12 Cardinality of a set

The cardinals are the initial ordinals.

lemma *Card-iff-initial*: *Card* $\kappa \longleftrightarrow$ *Ord* $\kappa \wedge (\forall \alpha.$ *Ord* $\alpha \wedge \alpha < \kappa \longrightarrow \sim$ *elts* $\alpha \approx$ *elts* κ)
  ⟨*proof*⟩

lemma *Card-ω* [*iff*]: *Card* ω
  ⟨*proof*⟩

lemma *lt-Card-imp-lesspoll*: $[\![i < a$; *Card a*; *Ord i*$]\!] \Longrightarrow$ *elts i* $\prec$ *elts a*
  ⟨*proof*⟩

lemma *lepoll-imp-Card-le*:
  assumes *elts a* $\lesssim$ *elts b* **shows** *vcard a* $\leq$ *vcard b*
  ⟨*proof*⟩

lemma *lepoll-cardinal-le*: $[\![$*elts A* $\lesssim$ *elts i*; *Ord i*$]\!] \Longrightarrow$ *vcard A* $\leq i$
  ⟨*proof*⟩

lemma *cardinal-le-lepoll*: *vcard A* $\leq \alpha \Longrightarrow$ *elts A* $\lesssim$ *elts* α
  ⟨*proof*⟩

lemma *lesspoll-imp-Card-less*:
  assumes *elts a* $\prec$ *elts b* **shows** *vcard a* $<$ *vcard b*
  ⟨*proof*⟩

lemma *Card-Union* [*simp*,*intro*]:
  assumes *A*: $\bigwedge x.\ x \in A \Longrightarrow$ *Card*(x) **shows** *Card*($\bigsqcup A$)
⟨*proof*⟩

lemma *Card-UN*: $(\bigwedge x.\ x \in A \Longrightarrow Card(K\ x)) ==> Card(Sup\ (K\ `\ A))$
$\langle proof \rangle$

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**
$transrec3\ a\ b\ c \equiv$
$transrec\ (\lambda r\ x.$
$if\ x{=}0\ then\ a$
$else\ if\ Limit\ x\ then\ c\ x\ (\lambda y \in elts\ x.\ r\ y)$
$else\ b(pred\ x)\ (r\ (pred\ x)))$

lemma *transrec3-0* [*simp*]: $transrec3\ a\ b\ c\ 0 = a$
$\langle proof \rangle$

lemma *transrec3-succ* [*simp*]:
$transrec3\ a\ b\ c\ (succ\ i) = b\ i\ (transrec3\ a\ b\ c\ i)$
$\langle proof \rangle$

lemma *transrec3-Limit* [*simp*]:
$Limit\ i \Longrightarrow\ transrec3\ a\ b\ c\ i = c\ i\ (\lambda j \in elts\ i.\ transrec3\ a\ b\ c\ j)$
$\langle proof \rangle$

2.14 Cardinal Addition

definition $cadd :: [V,V]\Rightarrow V$ (**infixl** ‹$\oplus$› *65*)
where $\kappa \oplus \mu \equiv vcard\ (\kappa \uplus \mu)$

2.14.1 Cardinal addition is commutative

lemma *vsum-commute-eqpoll*: $elts\ (a \uplus b) \approx elts\ (b \uplus a)$
$\langle proof \rangle$

lemma *cadd-commute*: $i \oplus j = j \oplus i$
$\langle proof \rangle$

2.14.2 Cardinal addition is associative

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

lemma *sum-assoc-eqpoll*: $elts\ ((a \uplus b) \uplus c) \approx elts\ (a \uplus (b \uplus c))$
$\langle proof \rangle$

lemma *elts-vcard-vsum-eqpoll*: $elts\ (vcard\ (i \uplus j)) \approx Inl\ `\ elts\ i \cup Inr\ `\ elts\ j$

⟨*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-eqpoll*: *elts* $(0 \uplus a) \approx$ *elts* a
⟨*proof*⟩

lemma *cadd-0* [*simp*]: *Card* $\kappa \Longrightarrow 0 \oplus \kappa = \kappa$
⟨*proof*⟩

lemma *cadd-0-right* [*simp*]: *Card* $\kappa \Longrightarrow \kappa \oplus 0 = \kappa$
⟨*proof*⟩

lemma *vsum-lepoll-self*: *elts* $a \lesssim$ *elts* $(a \uplus b)$
⟨*proof*⟩

lemma *cadd-le-self*:
assumes κ: *Card* κ **shows** $\kappa \le \kappa \oplus a$
⟨*proof*⟩

Monotonicity of addition

lemma *cadd-le-mono*: $[\![\kappa' \le \kappa; \mu' \le \mu]\!] \Longrightarrow \kappa' \oplus \mu' \le \kappa \oplus \mu$
⟨*proof*⟩

2.15 Cardinal multiplication

definition *cmult* :: $[V,V] \Rightarrow V$ (**infixl** ‹⊗› *70*)
where $\kappa \otimes \mu \equiv$ *vcard* (*VSigma* κ $(\lambda z.\ \mu)$)

2.15.1 Cardinal multiplication is commutative

lemma *prod-bij*: $[\![$*bij-betw f A C*; *bij-betw g B D*$]\!]$
$\Longrightarrow$ *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-eqpoll*: *elts* (*vcard* (*vtimes i j*)) $\approx$ *elts* $i \times$ *elts* j
⟨*proof*⟩

lemma *elts-cmult*: $elts\ (\kappa' \otimes \kappa) \approx elts\ \kappa' \times elts\ \kappa$
⟨*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*: $[\![\kappa' \leq \kappa;\ \mu' \leq \mu]\!] \Longrightarrow \kappa' \otimes \mu' \leq \kappa \otimes \mu$
⟨*proof*⟩

lemma *vcard-Sup-le-cmult*:
assumes $small\ U$ **and** κ: $\bigwedge x.\ x \in U \Longrightarrow vcard\ x \leq \kappa$
shows $vcard\ (\bigsqcup U) \leq vcard\ (set\ U) \otimes \kappa$
⟨*proof*⟩

2.17 The finite cardinals

lemma *succ-lepoll-succD*: $elts\ (succ(m)) \lesssim elts\ (succ(n)) \Longrightarrow elts\ m \lesssim elts\ n$
⟨*proof*⟩

Congruence law for *succ* under equipollence

lemma *succ-eqpoll-cong*: $elts\ a \approx elts\ b \Longrightarrow elts\ (succ(a)) \approx elts\ (succ(b))$
⟨*proof*⟩

lemma *sum-succ-eqpoll*: $elts\ (succ\ a \uplus b) \approx elts\ (succ(a \uplus b))$

⟨*proof*⟩

lemma *cadd-succ*: $succ\ m \oplus n = vcard\ (succ(m \oplus n))$
⟨*proof*⟩

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

lemma *vcard-disjoint-sup*:
assumes $x \sqcap y = 0$ **shows** $vcard\ (x \sqcup y) = vcard\ x \oplus vcard\ y$
⟨*proof*⟩

lemma *vcard-sup*: $vcard\ (x \sqcup y) \leq vcard\ x \oplus vcard\ y$
⟨*proof*⟩

2.18 Infinite cardinals

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

lemma *InfCard-iff*: $InfCard\ \kappa \longleftrightarrow Card\ \kappa \wedge infinite\ (elts\ \kappa)$
⟨*proof*⟩

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

lemma *InfCard-not-0*[*iff*]: $\neg\ InfCard\ 0$
⟨*proof*⟩

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

lemma *less-vcard-VPow*: $vcard\ A < vcard\ (VPow\ A)$
⟨*proof*⟩

lemma *greater-Card*:
assumes $Card\ \kappa$ **shows** $\kappa < vcard\ (VPow\ \kappa)$
⟨*proof*⟩

lemma
assumes $Card\ \kappa$
shows *Card-csucc* [*simp*]: $Card\ (csucc\ \kappa)$ **and** *less-csucc* [*simp*]: $\kappa < csucc\ \kappa$
⟨*proof*⟩

lemma *le-csucc*:
assumes $Card\ \kappa$ **shows** $\kappa \leq csucc\ \kappa$
⟨*proof*⟩

lemma *csucc-le*: $[\![Card\ \mu;\ \kappa \in elts\ \mu]\!] \Longrightarrow csucc\ \kappa \leq \mu$
⟨*proof*⟩

lemma *finite-csucc*: $a \in elts\ \omega \Longrightarrow csucc\ a = succ\ a$
⟨*proof*⟩

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

lemma *vcard-finite-set*: $finite\ A \Longrightarrow vcard\ (set\ A) = ord\text{-}of\text{-}nat\ (card\ A)$
⟨*proof*⟩

lemma *lt-csucc-iff*:
assumes *Ord* α *Card* κ
shows $\alpha < csucc\ \kappa \longleftrightarrow vcard\ \alpha \leq \kappa$
⟨*proof*⟩

lemma *Card-lt-csucc-iff*: $[\![Card\ \kappa';\ Card\ \kappa]\!] \Longrightarrow (\kappa' < csucc\ \kappa) = (\kappa' \leq \kappa)$
⟨*proof*⟩

lemma *csucc-lt-csucc-iff*: $[\![Card\ \kappa';\ Card\ \kappa]\!] \Longrightarrow (csucc\ \kappa' < csucc\ \kappa) = (\kappa' < \kappa)$
⟨*proof*⟩

lemma *csucc-le-csucc-iff*: $[\![Card\ \kappa';\ Card\ \kappa]\!] \Longrightarrow (csucc\ \kappa' \leq csucc\ \kappa) = (\kappa' \leq \kappa)$
⟨*proof*⟩

lemma *csucc-0* [*simp*]: $csucc\ 0 = 1$
⟨*proof*⟩

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

lemma *InfCard-csucc*: $InfCard\ \kappa \Longrightarrow InfCard\ (csucc\ \kappa)$
⟨*proof*⟩

Kunen's Lemma 10.11

lemma *InfCard-is-Limit*:
assumes *InfCard* κ **shows** *Limit* κ
⟨*proof*⟩

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

Kunen's Theorem 10.12

lemma *InfCard-csquare-eq*:

assumes $InfCard(\kappa)$ **shows** $\kappa \otimes \kappa = \kappa$
$\langle proof \rangle$

lemma *InfCard-le-cmult-eq*:
assumes $InfCard\ \kappa\ \mu \leq \kappa\ \mu \neq 0$
shows $\kappa \otimes \mu = \kappa$
$\langle proof \rangle$

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

lemma *InfCard-cmult-eq*: $[\![InfCard\ \kappa;\ InfCard\ \mu]\!] \Longrightarrow \kappa \otimes \mu = \kappa \sqcup \mu$
$\langle proof \rangle$

lemma *cmult-succ*:
$succ(m) \otimes n = n \oplus (m \otimes n)$
$\langle proof \rangle$

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

lemma *InfCard-cdouble-eq*:
assumes $InfCard\ \kappa$ **shows** $\kappa \oplus \kappa = \kappa$
$\langle proof \rangle$

Corollary 10.13 (1), for cardinal addition

lemma *InfCard-le-cadd-eq*: $[\![InfCard\ \kappa;\ \mu \leq \kappa]\!] \Longrightarrow \kappa \oplus \mu = \kappa$
$\langle proof \rangle$

lemma *InfCard-cadd-eq*: $[\![InfCard\ \kappa;\ InfCard\ \mu]\!] \Longrightarrow \kappa \oplus \mu = \kappa \sqcup \mu$
$\langle proof \rangle$

lemma *csucc-le-Card-iff*: $[\![Card\ \kappa';\ Card\ \kappa]\!] \Longrightarrow csucc\ \kappa' \leq \kappa \longleftrightarrow \kappa' < \kappa$
$\langle proof \rangle$

lemma *cadd-InfCard-le*:
assumes $\alpha \leq \kappa\ \beta \leq \kappa\ InfCard\ \kappa$
shows $\alpha \oplus \beta \leq \kappa$
$\langle proof \rangle$

lemma *cmult-InfCard-le*:
assumes $\alpha \leq \kappa\ \beta \leq \kappa\ InfCard\ \kappa$
shows $\alpha \otimes \beta \leq \kappa$
$\langle proof \rangle$

2.20 The Aleph-seqence

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

definition *Aleph* :: $V \Rightarrow V$ (‹ℵ-› [*90*] *90*)

where $Aleph \equiv transrec\ (\lambda f\ x.\ \omega \sqcup \bigsqcup((\lambda y.\ csucc(f\ y))\ `\ elts\ x))$

lemma *Aleph*: $Aleph\ \alpha = \omega \sqcup (\bigsqcup y{\in}elts\ \alpha.\ csucc\ (Aleph\ y))$
⟨*proof*⟩

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

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

lemma *Aleph-0* [*simp*]: $\aleph 0 = \omega$
⟨*proof*⟩

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

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

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

lemma *csucc-Aleph-le-Aleph*: $\alpha \in elts\ \beta \Longrightarrow csucc\ (\aleph\alpha) \leq \aleph\beta$
⟨*proof*⟩

lemma *Aleph-in-Aleph*: $\alpha \in elts\ \beta \Longrightarrow \aleph\alpha \in elts\ (\aleph\beta)$
⟨*proof*⟩

lemma *Aleph-Limit*:
assumes $Limit\ \gamma$
shows $Aleph\ \gamma = \bigsqcup\ (Aleph\ `\ elts\ \gamma)$
⟨*proof*⟩

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

lemma *countable-iff-le-Aleph0*: $countable\ (elts\ A) \longleftrightarrow vcard\ A \leq \aleph 0$
⟨*proof*⟩

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

lemma *vcard-Aleph* [*simp*]: $vcard\ (\aleph\alpha) = \aleph\alpha$
⟨*proof*⟩

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

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 $\omega 1 \equiv$ *Aleph 1*

lemma *Ord-ω1* [*simp*]: *Ord* $\omega 1$
⟨*proof*⟩

lemma *omega-ω1* [*iff*]: $\omega \in$ *elts* $\omega 1$
⟨*proof*⟩

lemma *ord-of-nat-ω1* [*iff*]: *ord-of-nat* $n \in$ *elts* $\omega 1$
⟨*proof*⟩

lemma *countable-iff-less-ω1*:
assumes *Ord* α
shows *countable* (*elts* α) $\longleftrightarrow \alpha < \omega 1$
⟨*proof*⟩

lemma *less-ω1-imp-countable*:
assumes $\alpha \in$ *elts* $\omega 1$
shows *countable* (*elts* α)
⟨*proof*⟩

lemma *ω1-gt0* [*simp*]: $\omega 1 > 0$
⟨*proof*⟩

lemma *ω1-gt1* [*simp*]: $\omega 1 > 1$
⟨*proof*⟩

lemma *Limit-ω1* [*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 $V :: \mathit{plus}$
begin

This definition is credited to Tarski

definition $\mathit{plus}\text{-}V :: V \Rightarrow V \Rightarrow V$
 where $\mathit{plus}\text{-}V\ x \equiv \mathit{transrec}\ (\lambda f\ z.\ x \sqcup \mathit{set}\ (f\ `\ \mathit{elts}\ z))$

instance $\langle \mathit{proof} \rangle$
end

definition $\mathit{lift} :: V \Rightarrow V \Rightarrow V$
 where $\mathit{lift}\ x\ y \equiv \mathit{set}\ (\mathit{plus}\ x\ `\ \mathit{elts}\ y)$

lemma plus: $x + y = x \sqcup \mathit{set}\ ((+)x\ `\ \mathit{elts}\ y)$
 $\langle \mathit{proof} \rangle$

lemma $\mathit{plus\text{-}eq\text{-}lift}$: $x + y = x \sqcup \mathit{lift}\ x\ y$
 $\langle \mathit{proof} \rangle$

Lemma 3.2

lemma $\mathit{lift\text{-}sup\text{-}distrib}$: $\mathit{lift}\ x\ (a \sqcup b) = \mathit{lift}\ x\ a \sqcup \mathit{lift}\ x\ b$
 $\langle \mathit{proof} \rangle$

lemma $\mathit{lift\text{-}Sup\text{-}distrib}$: $\mathit{small}\ Y \Longrightarrow \mathit{lift}\ x\ (\bigsqcup\ Y) = \bigsqcup\ (\mathit{lift}\ x\ `\ Y)$
 $\langle \mathit{proof} \rangle$

lemma $\mathit{add\text{-}Sup\text{-}distrib}$:
 fixes $x{::}V$ **shows** $y \neq 0 \Longrightarrow x + (\bigsqcup z{\in}\mathit{elts}\ y.\ f\ z) = (\bigsqcup z{\in}\mathit{elts}\ y.\ x + f\ z)$
 $\langle \mathit{proof} \rangle$

lemma $\mathit{Limit\text{-}add\text{-}Sup\text{-}distrib}$:
 fixes $x{::}V$ **shows** $\mathit{Limit}\ \alpha \Longrightarrow x + (\bigsqcup z{\in}\mathit{elts}\ \alpha.\ f\ z) = (\bigsqcup z{\in}\mathit{elts}\ \alpha.\ x + f\ z)$
 $\langle \mathit{proof} \rangle$

Proposition 3.3(ii)

instantiation $V :: \mathit{monoid\text{-}add}$
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 + vinsert z y = vinsert* (*x+z*) (*x + y*)
⟨*proof*⟩

lemma *plus-V-succ-right*: *x + succ y = succ* (*x + y*)
⟨*proof*⟩

lemma *succ-eq-add1*: *succ x = x + 1*
⟨*proof*⟩

lemma *ord-of-nat-add*: *ord-of-nat* (*m+n*) = *ord-of-nat m + ord-of-nat n*
⟨*proof*⟩

lemma *succ-0-plus-eq* [*simp*]:
assumes $\alpha \in$ *elts* ω
shows *succ 0* + α = *succ* α
⟨*proof*⟩

lemma *omega-closed-add* [*intro*]:
assumes $\alpha \in$ *elts* ω $\beta \in$ *elts* ω **shows** $\alpha+\beta \in$ *elts* ω
⟨*proof*⟩

lemma *mem-plus-V-E*:
assumes *l*: $l \in$ *elts* (*x + y*)
obtains $l \in$ *elts x* | *z* **where** $z \in$ *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* $\in$ *elts x*)
⟨*proof*⟩

Proposition 3.3(iii)

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

lemma *TC-sup-lift*: $TC\ x \sqcap lift\ x\ y = 0$
⟨*proof*⟩

lemma *lift-lift*: $lift\ x\ (lift\ y\ z) = lift\ (x{+}y)\ z$
⟨*proof*⟩

lemma *lift-self-disjoint*: $x \sqcap lift\ x\ u = 0$
⟨*proof*⟩

lemma *sup-lift-eq-lift*:
assumes $x \sqcup lift\ x\ u = x \sqcup lift\ x\ v$
shows $lift\ x\ u = lift\ x\ v$
⟨*proof*⟩

3.1.2 Deeper properties of addition

Proposition 3.4(i)

proposition *lift-eq-lift*: $lift\ x\ y = lift\ x\ z \Longrightarrow y = z$
⟨*proof*⟩

corollary *inj-lift*: $inj\text{-}on\ (lift\ x)\ A$
⟨*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** $x{+}y \in elts\ (x{+}z) \longleftrightarrow y \in elts\ z$
⟨*proof*⟩

corollary *add-le-cancel-left* [*iff*]:
fixes $x\ y\ z{::}V$ **shows** $x{+}y \le x{+}z \longleftrightarrow y{\le}z$
⟨*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*]: $lift\ x\ y \le x \longleftrightarrow y = 0$
⟨*proof*⟩

lemma *succ-less-ω-imp*: $succ\ x < \omega \Longrightarrow x < \omega$
⟨*proof*⟩

Proposition 3.5

lemma *card-lift*: *vcard* (*lift x y*) = *vcard y*
⟨*proof*⟩

lemma *eqpoll-lift*: *elts* (*lift x y*) ≈ *elts y*
⟨*proof*⟩

lemma *vcard-add*: *vcard* (*x* + *y*) = *vcard x* ⊕ *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* ⊔ *lift x* (*TC y*)
⟨*proof*⟩

corollary *TC-add'*: $z \sqsubset x + y \longleftrightarrow z \sqsubset x \lor (\exists v.\ v \sqsubset y \land z = x + v)$
⟨*proof*⟩

Corollary 3.7

corollary *vcard-TC-add*: *vcard* (*TC* (*x*+*y*)) = *vcard* (*TC x*) ⊕ *vcard* (*TC y*)
⟨*proof*⟩

Corollary 3.8

corollary *TC-lift*:
assumes $y \neq 0$
shows *TC* (*lift x y*) = *TC x* ⊔ *lift x* (*TC y*)
⟨*proof*⟩

proposition *rank-add-distrib*: *rank* (*x*+*y*) = *rank x* + *rank y*
⟨*proof*⟩

lemma *Ord-add* [*simp*]: ⟦*Ord x*; *Ord y*⟧ ⟹ *Ord* (*x*+*y*)
⟨*proof*⟩

lemma *add-Sup-distrib-id*: $A \neq 0 \Longrightarrow x + \bigsqcup(elts\ A) = (\bigsqcup z \in elts\ A.\ x + z)$
⟨*proof*⟩

lemma *add-Limit*: $Limit\ \alpha \Longrightarrow x + \alpha = (\bigsqcup z \in elts\ \alpha.\ x + z)$
⟨*proof*⟩

lemma *add-le-left*:
assumes *Ord* α *Ord* β **shows** $\beta \leq \alpha+\beta$
⟨*proof*⟩

lemma *plus-ω-equals-ω*:
assumes $\alpha \in elts\ \omega$ **shows** $\alpha + \omega = \omega$
⟨*proof*⟩

lemma *one-plus-ω-equals-ω* [*simp*]: $1 + \omega = \omega$
⟨*proof*⟩

3.1.3 Cancellation / set subtraction

definition *vle* :: $V \Rightarrow V \Rightarrow bool$ (**infix** ‹⊴› *50*)
where $x \trianglelefteq y \equiv \exists z{::}V.\ x{+}z = y$

lemma *vle-refl* [*iff*]: $x \trianglelefteq x$
⟨*proof*⟩

lemma *vle-antisym*: $[\![x \trianglelefteq y;\ y \trianglelefteq x]\!] \Longrightarrow x = y$
⟨*proof*⟩

lemma *vle-trans* [*trans*]: $[\![x \trianglelefteq y;\ y \trianglelefteq z]\!] \Longrightarrow x \trianglelefteq z$
⟨*proof*⟩

definition *vle-comparable* :: $V \Rightarrow V \Rightarrow bool$
where *vle-comparable* $x\ y \equiv x \trianglelefteq y \lor y \trianglelefteq x$

Lemma 3.13

lemma *comparable*:
assumes $a{+}b = c{+}d$
shows *vle-comparable* $a\ c$
⟨*proof*⟩

lemma *vle1*: $x \trianglelefteq y \Longrightarrow x \le y$
⟨*proof*⟩

lemma *vle2*: $x \trianglelefteq y \Longrightarrow x \sqsubseteq y$
⟨*proof*⟩

lemma *vle-iff-le-Ord*:
assumes *Ord* α *Ord* β
shows $\alpha \trianglelefteq \beta \longleftrightarrow \alpha \le \beta$
⟨*proof*⟩

lemma *add-le-cancel-left0* [*iff*]:
fixes $x{::}V$ **shows** $x \le x{+}z$
⟨*proof*⟩

lemma *add-less-cancel-left0* [*iff*]:
fixes $x{::}V$ **shows** $x < x{+}z \longleftrightarrow 0{<}z$
⟨*proof*⟩

lemma *le-Ord-diff*:
  assumes $\alpha \leq \beta$ *Ord* α *Ord* β
  obtains γ **where** $\alpha+\gamma = \beta$ $\gamma \leq \beta$ *Ord* γ
⟨*proof*⟩

lemma *plus-Ord-le*:
  assumes $\alpha \in$ *elts* ω *Ord* β **shows** $\alpha+\beta \leq \beta+\alpha$
⟨*proof*⟩

lemma *add-right-mono*: $[\![\alpha \leq \beta;$ *Ord* $\alpha;$ *Ord* $\beta;$ *Ord* $\gamma]\!] \Longrightarrow \alpha+\gamma \leq \beta+\gamma$
  ⟨*proof*⟩

lemma *add-strict-mono*: $[\![\alpha < \beta; \gamma < \delta;$ *Ord* $\alpha;$ *Ord* $\beta;$ *Ord* $\gamma;$ *Ord* $\delta]\!] \Longrightarrow \alpha+\gamma < \beta+\delta$
  ⟨*proof*⟩

lemma *add-right-strict-mono*: $[\![\alpha \leq \beta; \gamma < \delta;$ *Ord* $\alpha;$ *Ord* $\beta;$ *Ord* $\gamma;$ *Ord* $\delta]\!] \Longrightarrow \alpha+\gamma < \beta+\delta$
  ⟨*proof*⟩

lemma *Limit-add-Limit* [*simp*]:
  assumes *Limit* μ *Ord* β **shows** *Limit* $(\beta + \mu)$
  ⟨*proof*⟩

3.2 Generalised Difference

definition *odiff* **where** *odiff y x* $\equiv$ *THE z::V*. $(x+z = y) \vee (z=0 \wedge \neg\, x \trianglelefteq y)$

lemma *vle-imp-odiff-eq*: $x \trianglelefteq y \Longrightarrow x + ($*odiff y x*$) = y$
  ⟨*proof*⟩

lemma *not-vle-imp-odiff-0*: $\neg\, x \trianglelefteq y \Longrightarrow ($*odiff y x*$) = 0$
  ⟨*proof*⟩

lemma *Ord-odiff-eq*:
  assumes $\alpha \leq \beta$ *Ord* α *Ord* β
  shows $\alpha +$ *odiff* $\beta\ \alpha = \beta$
  ⟨*proof*⟩

lemma *Ord-odiff*:
  assumes *Ord* α *Ord* β **shows** *Ord* (*odiff* $\beta\ \alpha$)
⟨*proof*⟩

lemma *Ord-odiff-le*:
  assumes *Ord* α *Ord* β **shows** *odiff* $\beta\ \alpha \leq \beta$
⟨*proof*⟩

lemma *odiff-0-right* [*simp*]: *odiff x 0 = x*

⟨*proof*⟩

lemma *odiff-succ*: $y \trianglelefteq x \Longrightarrow odiff\ (succ\ x)\ y = succ\ (odiff\ x\ y)$
⟨*proof*⟩

lemma *odiff-eq-iff*: $z \trianglelefteq x \Longrightarrow odiff\ x\ z = y \longleftrightarrow x = z + y$
⟨*proof*⟩

lemma *odiff-le-iff*: $z \trianglelefteq x \Longrightarrow odiff\ x\ z \leq y \longleftrightarrow x \leq z + y$
⟨*proof*⟩

lemma *odiff-less-iff*: $z \trianglelefteq x \Longrightarrow odiff\ x\ z < y \longleftrightarrow x < z + y$
⟨*proof*⟩

lemma *odiff-ge-iff*: $z \trianglelefteq x \Longrightarrow odiff\ x\ z \geq y \longleftrightarrow x \geq z + y$
⟨*proof*⟩

lemma *Ord-odiff-le-iff*: $[\![\alpha \leq x;\ Ord\ x;\ Ord\ \alpha]\!] \Longrightarrow odiff\ x\ \alpha \leq y \longleftrightarrow x \leq \alpha + y$
⟨*proof*⟩

lemma *odiff-le-odiff*:
assumes $x \trianglelefteq y$ **shows** $odiff\ x\ z \leq odiff\ y\ z$
⟨*proof*⟩

lemma *Ord-odiff-le-odiff*: $[\![x \leq y;\ Ord\ x;\ Ord\ y]\!] \Longrightarrow odiff\ x\ \alpha \leq odiff\ y\ \alpha$
⟨*proof*⟩

lemma *Ord-odiff-less-odiff*: $[\![\alpha \leq x;\ x < y;\ Ord\ x;\ Ord\ y;\ Ord\ \alpha]\!] \Longrightarrow odiff\ x\ \alpha < odiff\ y\ \alpha$
⟨*proof*⟩

lemma *Ord-odiff-less-imp-less*: $[\![odiff\ x\ \alpha < odiff\ y\ \alpha;\ Ord\ x;\ Ord\ y]\!] \Longrightarrow x < y$
⟨*proof*⟩

lemma *odiff-add-cancel* [*simp*]: $odiff\ (x + y)\ x = y$
⟨*proof*⟩

lemma *odiff-add-cancel-0* [*simp*]: $odiff\ x\ x = 0$
⟨*proof*⟩

lemma *odiff-add-cancel-both* [*simp*]: $odiff\ (x + y)\ (x + z) = odiff\ y\ z$
⟨*proof*⟩

3.3 Generalised Multiplication

Credited to Dana Scott

instantiation *V* :: *times*
begin

This definition is credited to Tarski

definition *times-V* :: $V \Rightarrow V \Rightarrow V$
  where *times-V* $x \equiv$ *transrec* $(\lambda f\ y.\ \bigsqcup\ ((\lambda u.\ lift\ (f\ u)\ x)\ `\ elts\ y))$

instance ⟨*proof*⟩
end

lemma *mult*: $x * y = (\bigsqcup u{\in}elts\ y.\ lift\ (x * u)\ x)$
  ⟨*proof*⟩

lemma *elts-multE*:
  assumes $z \in elts\ (x * y)$
  obtains $u\ v$ **where** $u \in elts\ x\ v \in elts\ y\ z = x{*}v + u$
  ⟨*proof*⟩

Lemma 4.2

lemma *mult-zero-right* [*simp*]:
  fixes $x{::}V$ **shows** $x * 0 = 0$
  ⟨*proof*⟩

lemma *mult-insert*: $x * (vinsert\ y\ z) = x{*}z \sqcup lift\ (x{*}y)\ x$
  ⟨*proof*⟩

lemma *mult-succ*: $x * succ\ y = x{*}y + x$
  ⟨*proof*⟩

lemma *ord-of-nat-mult*: $ord\text{-}of\text{-}nat\ (m{*}n) = ord\text{-}of\text{-}nat\ m * ord\text{-}of\text{-}nat\ n$
⟨*proof*⟩

lemma *omega-closed-mult* [*intro*]:
  assumes $\alpha \in elts\ \omega\ \beta \in elts\ \omega$ **shows** $\alpha{*}\beta \in elts\ \omega$
⟨*proof*⟩

lemma *zero-imp-le-mult*: $0 \in elts\ y \Longrightarrow x \le x{*}y$
  ⟨*proof*⟩

3.3.1 Proposition 4.3

lemma *mult-zero-left* [*simp*]:
  fixes $x{::}V$ **shows** $0 * x = 0$
⟨*proof*⟩

lemma *mult-sup-distrib*:
  fixes $x{::}V$ **shows** $x * (y \sqcup z) = x{*}y \sqcup x{*}z$
  ⟨*proof*⟩

lemma *mult-Sup-distrib*: $small\ Y \Longrightarrow x * (\bigsqcup Y) = \bigsqcup\ ((*)\ x\ `\ Y)$ **for** $Y{::}\ V\ set$
  ⟨*proof*⟩

lemma *mult-lift-imp-distrib*: $x * (\mathit{lift}\ y\ z) = \mathit{lift}\ (x{*}y)\ (x{*}z) \Longrightarrow x * (y{+}z) = x{*}y + x{*}z$
⟨*proof*⟩

lemma *mult-lift*: $x * (\mathit{lift}\ y\ z) = \mathit{lift}\ (x{*}y)\ (x{*}z)$
⟨*proof*⟩

lemma *mult-Limit*: $\mathit{Limit}\ \gamma \Longrightarrow x * \gamma = \bigsqcup\ ((*)\ x\ `\ \mathit{elts}\ \gamma)$
⟨*proof*⟩

lemma *add-mult-distrib*: $x * (y{+}z) = x{*}y + x{*}z$ **for** $x{::}V$
⟨*proof*⟩

instantiation V :: *monoid-mult*
begin
instance
⟨*proof*⟩

end

lemma *le-mult*:
assumes $\mathit{Ord}\ \beta\ \beta \neq 0$ **shows** $\alpha \le \alpha * \beta$
⟨*proof*⟩

lemma *mult-sing-1* [*simp*]:
fixes $x{::}V$ **shows** $x * \mathit{set}\{1\} = \mathit{lift}\ x\ x$
⟨*proof*⟩

lemma *mult-2-right* [*simp*]:
fixes $x{::}V$ **shows** $x * \mathit{set}\{0,1\} = x{+}x$
⟨*proof*⟩

lemma *Ord-mult* [*simp*]: $[\![\mathit{Ord}\ y;\ \mathit{Ord}\ x]\!] \Longrightarrow \mathit{Ord}\ (x{*}y)$
⟨*proof*⟩

3.3.2 Proposition 4.4-5

proposition *rank-mult-distrib*: $\mathit{rank}\ (x{*}y) = \mathit{rank}\ x * \mathit{rank}\ y$
⟨*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** $x * y = 0 \longleftrightarrow x{=}0 \lor y{=}0$
⟨*proof*⟩

lemma *lift-lemma*:

assumes $x \neq 0$ $y \neq 0$ **shows** $\neg$ *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-ωE*:
assumes $x \in$ *elts* $(y * \omega)$
obtains n **where** $n \neq 0$ $x \in$ *elts* $(y *$ *ord-of-nat* $n)$ $\bigwedge m.\ m < n \Longrightarrow x \notin$ *elts* $(y$ $*$ *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 $x < y$ | $x = y$ $r < s$
⟨*proof*⟩

corollary *lift-mult-TC-disjoint*:
fixes x:: V
assumes $x \neq y$
shows *lift* $(a*x)$ $(TC\ a) \sqcap$ *lift* $(a*y)$ $(TC\ a) = 0$
⟨*proof*⟩

corollary *lift-mult-disjoint*:

fixes x:: V
assumes $x \neq y$
shows $\mathit{lift}\ (a{*}x)\ a \sqcap \mathit{lift}\ (a{*}y)\ a = 0$
$\langle \mathit{proof} \rangle$

lemma *mult-add-mem*:
assumes $a{*}x + r \in \mathit{elts}\ (a{*}y)\ r \sqsubset a$
shows $x \in \mathit{elts}\ y\ r \in \mathit{elts}\ a$
$\langle \mathit{proof} \rangle$

lemma *mult-add-mem-0* [*simp*]: $a{*}x \in \mathit{elts}\ (a{*}y) \longleftrightarrow x \in \mathit{elts}\ y \wedge 0 \in \mathit{elts}\ a$
$\langle \mathit{proof} \rangle$

lemma *zero-mem-mult-iff*: $0 \in \mathit{elts}\ (x{*}y) \longleftrightarrow 0 \in \mathit{elts}\ x \wedge 0 \in \mathit{elts}\ y$
$\langle \mathit{proof} \rangle$

lemma *zero-less-mult-iff* [*simp*]: $0 < x{*}y \longleftrightarrow 0 < x \wedge 0 < y$ **if** *Ord* x
$\langle \mathit{proof} \rangle$

lemma *mult-cancel-less-iff* [*simp*]:
$[\![\mathit{Ord}\ \alpha; \mathit{Ord}\ \beta; \mathit{Ord}\ \gamma]\!] \Longrightarrow \alpha{*}\beta < \alpha{*}\gamma \longleftrightarrow \beta < \gamma \wedge 0 < \alpha$
$\langle \mathit{proof} \rangle$

lemma *mult-cancel-le-iff* [*simp*]:
$[\![\mathit{Ord}\ \alpha; \mathit{Ord}\ \beta; \mathit{Ord}\ \gamma]\!] \Longrightarrow \alpha{*}\beta \leq \alpha{*}\gamma \longleftrightarrow \beta \leq \gamma \vee \alpha{=}0$
$\langle \mathit{proof} \rangle$

lemma *mult-Suc-add-less*: $[\![\alpha < \gamma; \beta < \gamma; \mathit{Ord}\ \alpha; \mathit{Ord}\ \beta; \mathit{Ord}\ \gamma]\!] \Longrightarrow \gamma * \mathit{ord\text{-}of\text{-}nat}\ m + \alpha < \gamma * \mathit{ord\text{-}of\text{-}nat}\ (\mathit{Suc}\ m) + \beta$
$\langle \mathit{proof} \rangle$

lemma *mult-nat-less-add-less*:
assumes $m < n\ \alpha < \gamma\ \beta < \gamma$ **and** *ord*: $\mathit{Ord}\ \alpha\ \mathit{Ord}\ \beta\ \mathit{Ord}\ \gamma$
shows $\gamma * \mathit{ord\text{-}of\text{-}nat}\ m + \alpha < \gamma * \mathit{ord\text{-}of\text{-}nat}\ n + \beta$
$\langle \mathit{proof} \rangle$

lemma *add-mult-less-add-mult*:
assumes $x < y\ x \in \mathit{elts}\ \beta\ y \in \mathit{elts}\ \beta\ \mu \in \mathit{elts}\ \alpha\ \nu \in \mathit{elts}\ \alpha\ \mathit{Ord}\ \alpha\ \mathit{Ord}\ \beta$
shows $\alpha{*}x + \mu < \alpha{*}y + \nu$
$\langle \mathit{proof} \rangle$

lemma *add-mult-less*:
assumes $\gamma \in \mathit{elts}\ \alpha\ \nu \in \mathit{elts}\ \beta\ \mathit{Ord}\ \alpha\ \mathit{Ord}\ \beta$
shows $\alpha * \nu + \gamma \in \mathit{elts}\ (\alpha * \beta)$
$\langle \mathit{proof} \rangle$

lemma *Ord-add-mult-iff*:
assumes $\beta \in \mathit{elts}\ \gamma\ \beta' \in \mathit{elts}\ \gamma\ \mathit{Ord}\ \alpha\ \mathit{Ord}\ \alpha'\ \mathit{Ord}\ \gamma$
shows $\gamma * \alpha + \beta \in \mathit{elts}\ (\gamma * \alpha' + \beta') \longleftrightarrow \alpha \in \mathit{elts}\ \alpha' \vee \alpha = \alpha' \wedge \beta \in \mathit{elts}\ \beta'$

(**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* α
  shows *ordertype* ((+) u ‘ *elts* α) *VWF* $= \alpha$
⟨*proof*⟩

lemma *ordertype-diff*:
  assumes $\beta + \delta = \alpha$ **and** α: $\delta \in$ *elts* α *Ord* α
  shows *ordertype* (*elts* $\alpha -$ *elts* β) *VWF* $= \delta$
⟨*proof*⟩

lemma *ordertype-interval-eq*:
  assumes α: *Ord* α **and** β: *Ord* β
  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* $(A{\times}B)$ (r <∗*lex*∗> s) $=$ *ordertype* B s ∗ *ordertype* A r (**is** - $=$ $?\beta * ?\alpha$)
⟨*proof*⟩

end

4 Exponentiation of ordinals

theory *Ordinal-Exp*
  imports *Kirby*

begin

Source: Schlöder, Julian. Ordinal Arithmetic; available online at http://www.math.uni-bonn.de/ag/logik/teaching/2012WS/Set%20theory/oa.pdf

definition *oexp* :: $[V,V] \Rightarrow V$ (**infixr** ‹↑› *80*)
  where *oexp a b* $\equiv$ *transrec* ($\lambda f\ x.$ *if* x=*0* *then* *1*
    else if Limit x then if a=0 then 0 else $\bigsqcup \xi \in$ *elts x. f* ξ
    else f $(\bigsqcup(\textit{elts } x)) * a)$ *b*

$0 \uparrow \omega = 1$ if we don't make a special case for Limit ordinals and zero

lemma *oexp-0-right* [*simp*]: $\alpha \uparrow 0 = 1$
  ⟨*proof*⟩

lemma *oexp-succ* [*simp*]: *Ord* $\beta \Longrightarrow \alpha \uparrow(\textit{succ } \beta) = \alpha \uparrow \beta * \alpha$
  ⟨*proof*⟩

lemma *oexp-Limit*: *Limit* $\beta \Longrightarrow \alpha \uparrow \beta = (\textit{if } \alpha{=}0 \textit{ then } 0 \textit{ else } \bigsqcup \xi \in \textit{elts } \beta.\ \alpha \uparrow \xi)$
  ⟨*proof*⟩

lemma *oexp-1-right* [*simp*]: $\alpha \uparrow 1 = \alpha$
  ⟨*proof*⟩

lemma *oexp-1* [*simp*]: *Ord* $\alpha \Longrightarrow 1 \uparrow \alpha = 1$
  ⟨*proof*⟩

lemma *oexp-0* [*simp*]: *Ord* $\alpha \Longrightarrow 0 \uparrow \alpha = (\textit{if } \alpha = 0 \textit{ then } 1 \textit{ else } 0)$
  ⟨*proof*⟩

lemma *oexp-eq-0-iff* [*simp*]:
  assumes *Ord* β **shows** $\alpha \uparrow \beta = 0 \longleftrightarrow \alpha{=}0 \wedge \beta{\neq}0$
  ⟨*proof*⟩

lemma *oexp-gt-0-iff* [*simp*]:
  assumes *Ord* β **shows** $\alpha \uparrow \beta > 0 \longleftrightarrow \alpha{>}0 \vee \beta{=}0$
  ⟨*proof*⟩

lemma *ord-of-nat-oexp*: *ord-of-nat* $(m \hat{\ } n)$ = *ord-of-nat m*↑*ord-of-nat n*
⟨*proof*⟩

lemma *omega-closed-oexp* [*intro*]:
  assumes $\alpha \in \textit{elts } \omega$ $\beta \in \textit{elts } \omega$ **shows** $\alpha \uparrow \beta \in \textit{elts } \omega$
⟨*proof*⟩

lemma *Ord-oexp* [*simp*]:
  assumes *Ord* α *Ord* β **shows** *Ord* $(\alpha \uparrow \beta)$
  ⟨*proof*⟩

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 β: $\beta \in elts\ \gamma$ **and** $Ord\ \gamma$ **and** α: $\alpha > 1\ Ord\ \alpha$ **shows** $\alpha\uparrow\beta < \alpha\uparrow\gamma$
⟨*proof*⟩

corollary *oexp-less-iff*:
  assumes $\alpha > 0\ Ord\ \alpha\ Ord\ \beta\ Ord\ \gamma$ **shows** $\alpha\uparrow\beta < \alpha\uparrow\gamma \longleftrightarrow \beta \in elts\ \gamma \wedge \alpha > 1$
⟨*proof*⟩

lemma *ω-oexp-iff* [*simp*]: $[\![Ord\ \alpha;\ Ord\ \beta]\!] \Longrightarrow \omega\uparrow\alpha = \omega\uparrow\beta \longleftrightarrow \alpha=\beta$
  ⟨*proof*⟩

lemma *Limit-oexp*:
  assumes $Limit\ \gamma\ Ord\ \alpha\ \alpha > 1$ **shows** $Limit\ (\alpha\uparrow\gamma)$
  ⟨*proof*⟩

lemma *oexp-mono*:
  assumes α: $Ord\ \alpha\ \alpha \neq 0$ **and** β: $Ord\ \beta\ \gamma \sqsubseteq \beta$ **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 α: $\alpha \neq 0\ Ord\ \alpha$ **and** X: $X \subseteq ON\ small\ X\ X \neq \{\}$ **shows** $\alpha\uparrow\bigsqcup X = \bigsqcup ((\uparrow)\ \alpha\ `\ X)$
⟨*proof*⟩

lemma *omega-le-Limit*:
  assumes *Limit* μ **shows** $\omega \leq \mu$
$\langle proof \rangle$

lemma *finite-omega-power* [*simp*]:
  assumes $1 < n$ $n \in$ *elts* ω **shows** $n \uparrow \omega = \omega$
$\langle proof \rangle$

proposition *oexp-add*:
  assumes *Ord* α *Ord* β *Ord* γ **shows** $\alpha \uparrow (\beta + \gamma) = \alpha \uparrow \beta * \alpha \uparrow \gamma$
$\langle proof \rangle$

proposition *oexp-mult*:
  assumes *Ord* α *Ord* β *Ord* γ **shows** $\alpha \uparrow (\beta * \gamma) = (\alpha \uparrow \beta) \uparrow \gamma$
$\langle proof \rangle$

lemma *Limit-omega-oexp*:
  assumes *Ord* δ $\delta \neq 0$
  shows *Limit* $(\omega \uparrow \delta)$
  $\langle proof \rangle$

lemma *oexp-mult-commute*:
  fixes *j*::*nat*
  assumes *Ord* α
  shows $(\alpha \uparrow j) * \alpha = \alpha * (\alpha \uparrow j)$
$\langle proof \rangle$

lemma *oexp-ω-Limit*: *Limit* $\beta \Longrightarrow \omega \uparrow \beta = (\bigsqcup \xi \in$ *elts* $\beta.\ \omega \uparrow \xi)$
  $\langle proof \rangle$

lemma *ω-power-succ-gtr*: *Ord* $\alpha \Longrightarrow \omega \uparrow \alpha *$ *ord-of-nat* $n < \omega \uparrow$ *succ* α
  $\langle proof \rangle$

lemma *countable-oexp*:
  assumes ν: $\alpha \in$ *elts* $\omega 1$
  shows $\omega \uparrow \alpha \in$ *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 α: $\alpha \in$ *elts* β *Ord* β **and** $m > 0$
  shows $\omega\uparrow\alpha *$ *ord-of-nat* $n < \omega\uparrow\beta *$ *ord-of-nat* m
⟨*proof*⟩

fun *Cantor-sum* **where**
  Cantor-sum-Nil: *Cantor-sum* [] *ms* = *0*
| *Cantor-sum-Nil2*: *Cantor-sum* (α#αs) [] = *0*
| *Cantor-sum-Cons*: *Cantor-sum* (α#αs) (m#*ms*) = ($\omega\uparrow\alpha$) $*$ *ord-of-nat* m + *Cantor-sum* αs *ms*

abbreviation *Cantor-dec* :: *V list* $\Rightarrow$ *bool* **where**
  Cantor-dec $\equiv$ *sorted-wrt* (>)

lemma *Ord-Cantor-sum*:
  assumes *List.set* $\alpha s \subseteq$ *ON*
  shows *Ord* (*Cantor-sum* αs *ms*)
  ⟨*proof*⟩

lemma *Cantor-dec-Cons-iff* [*simp*]: *Cantor-dec* (α#β#βs) $\longleftrightarrow$ $\beta < \alpha \wedge$ *Cantor-dec* (β#βs)
  ⟨*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 *List.set* (α#αs) $\subseteq$ *ON list.set ms* $\subseteq$ {*0*<..} *length* αs = *length ms*
    and *Cantor-dec* (α#αs)
  shows $\omega\uparrow\alpha >$ *Cantor-sum* αs *ms*
  ⟨*proof*⟩

proposition *Cantor-nf-exists*:
  assumes *Ord* α
  obtains αs *ms* **where** *List.set* $\alpha s \subseteq$ *ON list.set ms* $\subseteq$ {*0*<..} *length* αs = *length ms*
    and *Cantor-dec* αs
    and α = *Cantor-sum* αs *ms*
  ⟨*proof*⟩

lemma *Cantor-sum-0E*:
  assumes *Cantor-sum* αs *ms* = *0* *List.set* $\alpha s \subseteq$ *ON list.set ms* $\subseteq$ {*0*<..} *length* αs = *length ms*
  shows αs = []
  ⟨*proof*⟩

lemma *Cantor-nf-unique-aux*:
  assumes *Ord* α
    and αsON: *List.set* $\alpha s \subseteq ON$
    and βsON: *List.set* $\beta s \subseteq ON$
    and *ms*: *list.set ms* $\subseteq \{0<..\}$
    and *ns*: *list.set ns* $\subseteq \{0<..\}$
    and *mseq*: *length* αs = *length ms*
    and *nseq*: *length* βs = *length ns*
    and $\alpha sdec$: *Cantor-dec* αs
    and $\beta sdec$: *Cantor-dec* βs
    and αseq: α = *Cantor-sum* αs *ms*
    and βseq: α = *Cantor-sum* βs *ns*
  shows $\alpha s = \beta s \wedge ms = ns$
  ⟨*proof*⟩

proposition *Cantor-nf-unique*:
  assumes *Cantor-sum* αs *ms* = *Cantor-sum* βs *ns*
    and αsON: *List.set* $\alpha s \subseteq ON$
    and βsON: *List.set* $\beta s \subseteq ON$
    and *ms*: *list.set ms* $\subseteq \{0<..\}$
    and *ns*: *list.set ns* $\subseteq \{0<..\}$
    and *mseq*: *length* αs = *length ms*
    and *nseq*: *length* βs = *length ns*
    and $\alpha sdec$: *Cantor-dec* αs
    and $\beta sdec$: *Cantor-dec* βs
  shows $\alpha s = \beta s \wedge ms = ns$
  ⟨*proof*⟩

lemma *less-*ω*-power*:
  assumes *Ord* $\alpha 1$ *Ord* β
    and $\alpha 2$: $\alpha 2 \in$ *elts* $\alpha 1$ **and** β: $\beta < \omega\uparrow\alpha 2$
    and $m1 > 0$ $m2 > 0$
  shows $\omega\uparrow\alpha 2 *$ *ord-of-nat* $m2 + \beta < \omega\uparrow\alpha 1 *$ *ord-of-nat* $m1 + (\omega\uparrow\alpha 2 *$ *ord-of-nat* $m2 + \beta)$
        (**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 $\omega \uparrow \alpha \leq$ *Cantor-sum* $(\alpha\#\alpha s)$ *ms*
⟨*proof*⟩

5.2 Simplified Cantor normal form

No coefficients, and the exponents decreasing non-strictly

fun ω-*sum* **where**
ω-*sum-Nil*: ω-*sum* [] = *0*
| ω-*sum-Cons*: ω-*sum* $(\alpha\#\alpha s) = (\omega\uparrow\alpha) + \omega$-*sum* αs

abbreviation ω-*dec* :: *V list* $\Rightarrow$ *bool* **where**
ω-*dec* $\equiv$ *sorted-wrt* $(\geq)$

lemma *Ord*-ω-*sum* [*simp*]: *List.set* $\alpha s \subseteq ON \Longrightarrow$ *Ord* (ω-*sum* αs)
⟨*proof*⟩

lemma ω-*dec-Cons-iff* [*simp*]: ω-*dec* $(\alpha\#\beta\#\beta s) \longleftrightarrow \beta \leq \alpha \wedge \omega$-*dec* $(\beta\#\beta s)$
⟨*proof*⟩

lemma ω-*sum-0E*:
assumes ω-*sum* $\alpha s = 0$ *List.set* $\alpha s \subseteq ON$
shows $\alpha s = []$
⟨*proof*⟩

fun ω-*of-Cantor* **where**
ω-*of-Cantor-Nil*: ω-*of-Cantor* [] *ms* = []
| ω-*of-Cantor-Nil2*: ω-*of-Cantor* $(\alpha\#\alpha s)$ [] = []
| ω-*of-Cantor-Cons*: ω-*of-Cantor* $(\alpha\#\alpha s)$ $(m\#ms)$ = *replicate* m α @ ω-*of-Cantor* αs *ms*

lemma ω-*sum-append* [*simp*]: ω-*sum* (*xs* @ *ys*) = ω-*sum* *xs* + ω-*sum* *ys*
⟨*proof*⟩

lemma ω-*sum-replicate* [*simp*]: ω-*sum* (*replicate* m a) = $\omega \uparrow a *$ *ord-of-nat* m
⟨*proof*⟩

lemma ω-*sum-of-Cantor* [*simp*]: ω-*sum* (ω-*of-Cantor* αs *ms*) = *Cantor-sum* αs *ms*
⟨*proof*⟩

lemma ω-*of-Cantor-subset*: *List.set* (ω-*of-Cantor* αs *ms*) $\subseteq$ *List.set* αs
⟨*proof*⟩

lemma ω-*dec-replicate*: ω-*dec* (*replicate* m α @ αs) = (*if* m=*0* *then* ω-*dec* αs *else* ω-*dec* $(\alpha\#\alpha s)$)
⟨*proof*⟩

lemma ω-*dec-of-Cantor-aux*:
assumes *Cantor-dec* $(\alpha\#\alpha s)$ *length* αs = *length* *ms*
shows ω-*dec* (ω-*of-Cantor* $(\alpha\#\alpha s)$ $(m\#ms)$)
⟨*proof*⟩

lemma ω*-dec-of-Cantor*:
  assumes *Cantor-dec* αs *length* αs = *length ms*
  shows ω*-dec* (ω*-of-Cantor* αs *ms*)
⟨*proof*⟩

proposition ω*-nf-exists*:
  assumes *Ord* α
  obtains αs **where** *List.set* $\alpha s \subseteq ON$ **and** ω*-dec* αs **and** $\alpha = \omega$*-sum* αs
⟨*proof*⟩

lemma ω*-sum-take-drop*: ω*-sum* $\alpha s = \omega$*-sum* (*take k* αs) + ω*-sum* (*drop k* αs)
⟨*proof*⟩

lemma *in-elts-*ω*-sum*:
  assumes $\delta \in$ *elts* (ω*-sum* αs)
  shows $\exists k<$*length* αs. $\exists \gamma\in$*elts* $(\omega \uparrow (\alpha s!k))$. $\delta = \omega$*-sum* (*take k* αs) + γ
  ⟨*proof*⟩

lemma ω*-le-*ω*-sum*: $[\![k <$ *length* αs; *List.set* $\alpha s \subseteq ON]\!] \Longrightarrow \omega \uparrow (\alpha s!k) \leq \omega$*-sum* αs
⟨*proof*⟩

lemma ω*-sum-less-self*:
  assumes *List.set* $(\alpha\#\alpha s) \subseteq ON$ **and** ω*-dec* $(\alpha\#\alpha s)$
  shows ω*-sum* $\alpha s < \omega\uparrow\alpha + \omega$*-sum* αs
  ⟨*proof*⟩

Something like Lemma 5.2 for ω*-sum*

lemma ω*-sum-less-*ω*-power*:
  assumes ω*-dec* $(\alpha\#\alpha s)$ *List.set* $(\alpha\#\alpha s) \subseteq ON$
  shows ω*-sum* $\alpha s < \omega\uparrow\alpha * \omega$
  ⟨*proof*⟩

lemma ω*-sum-nf-unique-aux*:
  assumes *Ord* α
    and αsON: *List.set* $\alpha s \subseteq ON$
    and βsON: *List.set* $\beta s \subseteq ON$
    and $\alpha sdec$: ω*-dec* αs
    and $\beta sdec$: ω*-dec* βs
    and αseq: $\alpha = \omega$*-sum* αs
    and βseq: $\alpha = \omega$*-sum* βs
  shows $\alpha s = \beta s$
  ⟨*proof*⟩

5.3 Indecomposable ordinals

Cf exercise 5 on page 43 of Kunen

definition *indecomposable*

where *indecomposable* $\alpha \equiv$ *Ord* $\alpha \wedge (\forall \beta \in$ *elts* $\alpha.\ \forall \gamma \in$ *elts* $\alpha.\ \beta+\gamma \in$ *elts* $\alpha)$

lemma *indecomposableD*:
$[\![$*indecomposable* α; $\beta < \alpha$; $\gamma < \alpha$; *Ord* β; *Ord* $\gamma]\!] \Longrightarrow \beta+\gamma < \alpha$
⟨*proof*⟩

lemma *indecomposable-imp-Ord*:
indecomposable $\alpha \Longrightarrow$ *Ord* α
⟨*proof*⟩

lemma *indecomposable-1*: *indecomposable 1*
⟨*proof*⟩

lemma *indecomposable-0*: *indecomposable 0*
⟨*proof*⟩

lemma *indecomposable-succ* [*simp*]: *indecomposable* (*succ* α) $\longleftrightarrow \alpha = 0$
⟨*proof*⟩

lemma *indecomposable-alt*:
assumes *ord*: *Ord* α *Ord* β **and** β: $\beta < \alpha$ **and** *minor*: $\bigwedge\beta\ \gamma.\ [\![\beta < \alpha;\ \gamma < \alpha;$ *Ord* $\gamma]\!] \Longrightarrow \beta+\gamma < \alpha$
shows $\beta+\alpha = \alpha$
⟨*proof*⟩

lemma *indecomposable-imp-eq*:
assumes *indecomposable* α *Ord* β $\beta < \alpha$
shows $\beta+\alpha = \alpha$
⟨*proof*⟩

lemma *indecomposable2*:
assumes *y*: $y < x$ **and** *z*: $z < x$ **and** *minor*: $\bigwedge y{::}V.\ y < x \Longrightarrow y+x = x$
shows $y+z < x$
⟨*proof*⟩

lemma *indecomposable-imp-Limit*:
assumes *indec*: *indecomposable* α **and** $\alpha > 1$
shows *Limit* α
⟨*proof*⟩

lemma *eq-imp-indecomposable*:
assumes *Ord* α $\bigwedge\beta{::}V.\ \beta \in$ *elts* $\alpha \Longrightarrow \beta+\alpha = \alpha$
shows *indecomposable* α
⟨*proof*⟩

lemma *indecomposable-ω-power*:
assumes *Ord* δ
shows *indecomposable* $(\omega\uparrow\delta)$
⟨*proof*⟩

lemma ω*-power-imp-eq*:
assumes $\beta < \omega\uparrow\delta$ *Ord* β *Ord* δ $\delta \neq 0$
shows $\beta + \omega\uparrow\delta = \omega\uparrow\delta$
⟨*proof*⟩

lemma *mult-oexp-indec*: $[\![$*Ord* α; *Limit* μ; *indecomposable* $\mu]\!] \Longrightarrow \alpha * (\alpha \uparrow \mu) = (\alpha \uparrow \mu)$
⟨*proof*⟩

lemma *mult-oexp-*ω: *Ord* $\alpha \Longrightarrow \alpha * (\alpha \uparrow \omega) = (\alpha \uparrow \omega)$
⟨*proof*⟩

lemma *type-imp-indecomposable*:
assumes α: *Ord* α
and *minor*: $\bigwedge X.\ X \subseteq$ *elts* $\alpha \Longrightarrow$ *ordertype* X *VWF* $= \alpha \vee$ *ordertype* (*elts* $\alpha - X$) *VWF* $= \alpha$
shows *indecomposable* α
⟨*proof*⟩

This proof uses Cantor normal form, yet still is rather long

proposition *indecomposable-is-*ω*-power*:
assumes *inc*: *indecomposable* μ
obtains $\mu = 0 \mid \delta$ **where** *Ord* δ $\mu = \omega\uparrow\delta$
⟨*proof*⟩

corollary *indecomposable-iff-*ω*-power*:
indecomposable $\mu \longleftrightarrow \mu = 0 \vee (\exists\,\delta.\ \mu = \omega\uparrow\delta \wedge$ *Ord* $\delta)$
⟨*proof*⟩

theorem *indecomposable-imp-type*:
fixes X :: *bool* $\Rightarrow$ *V set*
assumes γ: *indecomposable* γ
and $\bigwedge b.$ *ordertype* $(X\ b)$ *VWF* $\leq \gamma$ $\bigwedge b.$ *small* $(X\ b)$ $\bigwedge b.\ X\ b \subseteq ON$
and *elts* $\gamma \subseteq (UN\ b.\ X\ b)$
shows $\exists\, b.$ *ordertype* $(X\ b)$ *VWF* $= \gamma$
⟨*proof*⟩

corollary *indecomposable-imp-type2*:
assumes α: *indecomposable* γ $X \subseteq$ *elts* γ
shows *ordertype* X *VWF* $= \gamma \vee$ *ordertype* (*elts* $\gamma - X$) *VWF* $= \gamma$
⟨*proof*⟩

5.4 From ordinals to order types

lemma *indecomposable-ordertype-eq*:
assumes *indec*: *indecomposable* α **and** α: *ordertype* A *VWF* $= \alpha$ **and** A: $B \subseteq A$ *small* A
shows *ordertype* B *VWF* $= \alpha \vee$ *ordertype* $(A{-}B)$ *VWF* $= \alpha$

$\langle proof \rangle$

lemma *indecomposable-ordertype-ge*:
assumes *indec*: *indecomposable* α **and** α: *ordertype* A *VWF* $\geq \alpha$ **and** *small*: *small* A *small* B
shows *ordertype* B *VWF* $\geq \alpha \vee$ *ordertype* $(A-B)$ *VWF* $\geq \alpha$
$\langle proof \rangle$

now for finite partitions

lemma *indecomposable-ordertype-finite-eq*:
assumes *indecomposable* α
and $\mathcal{A}$: *finite* $\mathcal{A}$ *pairwise disjnt* $\mathcal{A}$ $\bigcup\mathcal{A} = A$ $\mathcal{A} \neq \{\}$ *ordertype* A *VWF* $= \alpha$ *small* A
shows $\exists X \in \mathcal{A}$. *ordertype* X *VWF* $= \alpha$
$\langle proof \rangle$

lemma *indecomposable-ordertype-finite-ge*:
assumes *indec*: *indecomposable* α
and $\mathcal{A}$: *finite* $\mathcal{A}$ $A \subseteq \bigcup\mathcal{A}$ $\mathcal{A} \neq \{\}$ *ordertype* A *VWF* $\geq \alpha$ *small* $(\bigcup\mathcal{A})$
shows $\exists X \in \mathcal{A}$. *ordertype* X *VWF* $\geq \alpha$
$\langle proof \rangle$

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-of* :: *'a* $\Rightarrow$ *V*. *inj V-of*

context *countable*
begin

subclass *embeddable*
$\langle proof \rangle$

end

instance *unit* :: *embeddable* $\langle proof \rangle$
instance *bool* :: *embeddable* $\langle proof \rangle$
instance *nat* :: *embeddable* $\langle proof \rangle$
instance *int* :: *embeddable* $\langle proof \rangle$
instance *rat* :: *embeddable* $\langle proof \rangle$

instance *char* :: *embeddable* ⟨*proof*⟩
instance *String.literal* :: *embeddable* ⟨*proof*⟩
instance *typerep* :: *embeddable* ⟨*proof*⟩

lemma *embeddable-classI*:
  fixes f :: $'a \Rightarrow V$
  assumes $\bigwedge x\ y.\ f\ x = f\ y \Longrightarrow x = y$
  shows $OFCLASS('a,\ embeddable\text{-}class)$
⟨*proof*⟩

instance *V* :: *embeddable*
  ⟨*proof*⟩

instance *prod* :: (*embeddable*,*embeddable*) *embeddable*
⟨*proof*⟩

instance *sum* :: (*embeddable*,*embeddable*) *embeddable*
⟨*proof*⟩

instance *option* :: (*embeddable*) *embeddable*
⟨*proof*⟩

primrec *V-of-list* **where**
  V-of-list V-of Nil = *0*
| *V-of-list V-of* (*x#xs*) = ⟨*V-of x*, *V-of-list V-of xs*⟩

lemma *inj-V-of-list*:
  assumes *inj V-of*
  shows *inj* (*V-of-list V-of*)
⟨*proof*⟩

instance *list* :: (*embeddable*) *embeddable*
⟨*proof*⟩

6.2 The class of small types

class *small* =
  assumes *small*: *small* (*UNIV*::*'a set*)
begin

subclass *embeddable*
  ⟨*proof*⟩

lemma *TC-small* [*iff*]:
  fixes *A* :: *'a set*
  shows *small A*
  ⟨*proof*⟩

end

context *countable*
begin

subclass *small*
⟨*proof*⟩

end

lemma *lepoll-UNIV-imp-small*: $X \lesssim (UNIV::'a::small\ set) \Longrightarrow small\ X$
⟨*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 ℝ and ℂ and the cardinality of the contimuum

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* ⇒ *V*
where *V-of* ≡ *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*) ⟹ *infinite* (*range* (*V-of*::*'a*::*embeddable*⇒*V*))
⟨*proof*⟩

lemma *countable-V-of*: *countable* (*range* (*V-of*::*'a*::*countable*⇒*V*))
⟨*proof*⟩

lemma *elts-set-V-of*: *small X* ⟹ *elts* (*ZFC-in-HOL.set* (*V-of* ‘ *X*)) ≈ *X*
⟨*proof*⟩

lemma *V-of-image-times*: *V-of* ‘ (*X* × *Y*) ≈ (*V-of* ‘ *X*) × (*V-of* ‘ *Y*)
⟨*proof*⟩

7.2 The cardinality of the continuum

definition *Real-set* ≡ *ZFC-in-HOL.set* (*range* (*V-of*::*real*⇒*V*))
definition *Complex-set* ≡ *ZFC-in-HOL.set* (*range* (*V-of*::*complex*⇒*V*))
definition *C-continuum* ≡ *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* $\geq$ ℵ*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** κ: $\bigwedge x.\ x \in U \Longrightarrow$ *gcard x* $\leq \kappa$ **and** *sm*: $\bigwedge x.\ x \in U \Longrightarrow$ *small x*
shows *gcard* ($\bigcup$ *U*) $\leq$ *gcard U* $\otimes$ κ
⟨*proof*⟩

lemma *gcard-Times* [*simp*]: *gcard* (*X* $\times$ *Y*) = *gcard X* $\otimes$ *gcard Y*
⟨*proof*⟩

7.3 Countable and uncountable sets

lemma *countable-iff-g-le-Aleph0*:
assumes *small X*
shows *countable X* $\longleftrightarrow$ *gcard X* $\leq$ ℵ*0*
⟨*proof*⟩

lemma *countable-imp-g-le-Aleph0*: *countable X* $\Longrightarrow$ *gcard X* $\leq$ ℵ*0*
⟨*proof*⟩

lemma *finite-iff-g-le-Aleph0*: *small X* $\Longrightarrow$ *finite X* $\longleftrightarrow$ *gcard X* < ℵ*0*
⟨*proof*⟩

lemma *finite-imp-g-le-Aleph0*: *finite X* $\Longrightarrow$ *gcard X* < ℵ*0*
⟨*proof*⟩

lemma *countable-infinite-gcard*: *countable X* $\wedge$ *infinite X* $\longleftrightarrow$ *gcard X* = ℵ*0*
⟨*proof*⟩

lemma *uncountable-gcard*: *small X* $\Longrightarrow$ *uncountable X* $\longleftrightarrow$ *gcard X* > ℵ*0*
⟨*proof*⟩

lemma *uncountable-gcard-ge*: *small X* $\Longrightarrow$ *uncountable X* $\longleftrightarrow$ *gcard X* $\geq$ $\aleph$*1*
⟨*proof*⟩

lemma *subset-smaller-gcard*:
assumes κ: $\kappa \leq$ *gcard X Card* κ
obtains *Y* **where** *Y* $\subseteq$ *X gcard Y* $= \kappa$
⟨*proof*⟩

lemma *Real-gcard*: *gcard* (*UNIV*::*real set*) = *C-continuum*
⟨*proof*⟩

lemma *Complex-gcard*: *gcard* (*UNIV*::*complex set*) = *C-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.