Ordinals and cardinals in HOL

Andrei Popescu & Dmitriy Traytel

Contents

Abstract

We develop a basic theory of ordinals and cardinals in Isabelle/HOL, up to the point where some cardinality facts relevant for the "working mathematician" become available. Unlike in set theory, here we do not have at hand canonical notions of ordinal and cardinal. Therefore, here an ordinal is merely a well-order relation and a cardinal is an ordinal minim w.r.t. order embedding on its field.

1 Introduction

In set theory (under formalizations such as Zermelo-Fraenkel or Von Neumann-Bernays-Gödel), an *ordinal* is a special kind of well-order, namely one whose strict version is the restriction of the membership relation to a set. In particular, the field of a set-theoretic ordinal is a transitive set, and the non-strict version of an ordinal relation is set inclusion. Set-theoretic ordinals enjoy the nice properties of membership on transitive sets, while at the same time forming a complete class of representatives for well-orders (since any well-order turns out isomorphic to an ordinal). Moreover, the class of ordinals is itself transitive and well-ordered by membership as the strict relation and inclusion as the non-strict relation. Also knowing that any set can be well-ordered (in the presence of the axiom of choice), one then defines the *cardinal* of a set to be the smallest ordinal isomorphic to a well-order on that set. This makes the class of cardinals a complete set of representatives for the intuitive notion of set cardinality.[1] The ability to produce *canonical well-orders* from the membership relation (having the aforementioned convenient properties) allows for a harmonious development of the theory of cardinals in set-theoretic settings. Non-trivial cardinality results, such as A being equipollent to $A \times A$ for any infinite A, follow rather quickly within this theory.

However, a canonical notion of well-order is *not* available in HOL. Here, one has to do with well-order "as is", but otherwise has all the necessary infrastructure (including Hilbert choice) to "climb" well-orders recursively and to well-oder arbitrary sets.

The current work, formalized in Isabelle/HOL, develops the basic theory of ordinals and cardinals up to the point where there are inferred a collection of non-trivial cardinality facts useful for the "working mathematician", among which:

[1] The "intuitive" cardinality of a set A is the class of all sets equipollent to A, i.e., being in bijection with A.

- Given any two sets (on any two given types)[2], one is injectable in the other.
- If at least one of two sets is infinite, then their sum and their Cartesian product are equipollent to the larger of the two.
- The set of lists (and also the set of finite sets) with element from an infinite set is equipollent with that set.

Our development emulates the standard one from set-theory, but keeps everything *up to order isomorphism.* An (HOL) ordinal is merely a well-order. An *ordinal embedding* is an injective and order-compatible function which maps its source into an initial segment (i.e., order filter) of its target. Now, a *cardinal* (called in this work a *cardinal order*) is an ordinal minim w.r.t. the existence of embeddings among all well-orders on its field. After showing the existence of cardinals on any given set, we define the cardinal of a set A, denoted $|A|$, to be *some* cardinal order on A. This concept is unique only up to order isomorphism (denoted $=o$), but meets its purpose: any two sets A and B (laying at potentially distinct types) are in bijection if and only if $|A| =o |B|$. Moreover, we also show that numeric cardinals assigned to finite sets[3] are *conservatively extended* by our general (order-theoretic) notion of cardinal. We study the interaction of cardinals with standard set-theoretic constructions such as powersets, products, sums and lists. These constructions are shown to preserve the "cardinal identity" $=o$ and also to be monotonic w.r.t. $\leq o$, the ordinal embedding relation. By studying the interaction between these constructions, infinite sets and cardinals, we obtain the aforementioned results for "working mathematicians".

For this development, we did not follow closely any particular textbook, and in fact are not aware of such basic theory of cardinals previously developed in HOL.[4] On the other hand, the set-theoretic versions of the facts proved here are folklore in set theory, and can be found, e.g., in the textbook [1]. Beyond taking care of some locality aspects concerning the spreading of our concepts throughout types, we have not departed much from the techniques used in set theory for establishing these facts – for instance, in the proof of one of our major theorems, *Card-order-Times-same-infinite* from Section 8.4,[5] we have essentially applied the technique described, e.g., in the proof of theorem 1.5.11 from [1], page 47.

Here is the structure of the rest of this document.

[2]Recall that, in HOL, a set on a type α is modeled, just like a predicate, as a function from α to bool.

[3]Numeric cardinals of finite sets are already formalized in Isabelle/HOL.

[4]After writing this formalization, we became aware of Paul Taylor's membership-free development of the theory of ordinals [2].

[5]This theorem states that, for any infinite cardinal r on a set A, $|A \times A|$ is not larger than r.

The next three sections, 2-4, develop some mathematical prerequisites. In Section 2, a large collection of simple facts about injections, bijections, inverses, (in)finite sets and numeric cardinals are proved, making life easier for later, when proving less trivial facts. Section 3 introduces upper and lower bounds operators for order-like relations and studies their basic properties. Section 4 states some useful variations of well-founded recursion and induction principles.

Then come the major sections, 5-8. Section 5 defines and studies, in the context of a well-order relation, the notions of minimum (of a set), maximum (of two elements), supremum, successor (of a set), and order filter (i.e., initial segment, i.e., downward-closed set). Section 6 defines and studies (well-order) embeddings, strict embeddings, isomorphisms, and compatible functions. Section 7 deals with various constructions on well-orders, and with the relations $\leq o$, $<o$ and $=o$ of well-order embedding, strict embedding, and isomorphism, respectively. Section 8 defines and studies cardinal order relations, the cardinal of a set, the connection of cardinals with set-theoretic constructs, the canonical cardinal of natural numbers and finite cardinals, the successor of a cardinal, as well as regular cardinals. (The latter play a crucial role in the development of a new (co)datatype package in HOL.)

Finally, section 9 provides an abstraction of the previous developments on cardinals, to provide a simpler user interface to cardinals, which in most of the cases allows to forget that cardinals are represented by orders and use them through defined arithmetic operators.

More informal details are provided at the beginning of each section, and also at the beginning of some of the subsections.

References

[1] M. Holz, K. Steffens, and E. Weitz. *Introduction to Cardinal Arithmetic*. Birkhäuser, 1999.

[2] Paul Taylor. Intuitionistic sets and ordinals. *J. Symb. Log.*, 61(3):705–744, 1996.

2 More on Injections, Bijections and Inverses

theory *Fun-More*
  imports *Main*
begin

2.1 Purely functional properties

lemma *bij-betw-diff-singl*:
  assumes *BIJ*: *bij-betw f A A′* **and** *IN*: $a \in A$

shows *bij-betw f* $(A - \{a\})$ $(A' - \{f\ a\})$
⟨*proof*⟩

2.2 Properties involving finite and infinite sets

lemma *bij-betw-inv-into-RIGHT*:
assumes *BIJ*: *bij-betw f A A'* **and** *SUB*: $B' \leq A'$
shows f ‘((*inv-into A f*)‘B') $= B'$
⟨*proof*⟩

lemma *bij-betw-inv-into-RIGHT-LEFT*:
assumes *BIJ*: *bij-betw f A A'* **and** *SUB*: $B' \leq A'$ **and**
IM: (*inv-into A f*) ‘ $B' = B$
shows f ‘ $B = B'$
⟨*proof*⟩

lemma *bij-betw-inv-into-twice*:
assumes *bij-betw f A A'*
shows $\forall\, a \in A.$ *inv-into A'* (*inv-into A f*) $a = f\ a$
⟨*proof*⟩

2.3 Properties involving Hilbert choice

lemma *bij-betw-inv-into-LEFT*:
assumes *BIJ*: *bij-betw f A A'* **and** *SUB*: $B \leq A$
shows (*inv-into A f*)‘(f ‘ B) $= B$
⟨*proof*⟩

lemma *bij-betw-inv-into-LEFT-RIGHT*:
assumes *BIJ*: *bij-betw f A A'* **and** *SUB*: $B \leq A$ **and**
IM: f ‘ $B = B'$
shows (*inv-into A f*) ‘ $B' = B$
⟨*proof*⟩

2.4 Other facts

lemma *atLeastLessThan-injective*:
assumes $\{0\ ..< m::nat\} = \{0\ ..< n\}$
shows $m = n$
⟨*proof*⟩

lemma *atLeastLessThan-injective2*:
bij-betw f $\{0\ ..< m::nat\}$ $\{0\ ..< n\} \Longrightarrow m = n$
⟨*proof*⟩

lemma *atLeastLessThan-less-eq*:
$(\{0..<m\} \leq \{0..<n\}) = ((m::nat) \leq n)$

$\langle proof \rangle$

lemma *atLeastLessThan-less-eq2*:
  assumes $inj\text{-}on\ f\ \{0..<(m::nat)\}\ f\ `\ \{0..<m\} \leq \{0..<n\}$
  shows $m \leq n$
  $\langle proof \rangle$

lemma *atLeastLessThan-less*:
  $(\{0..<m\} < \{0..<n\}) = ((m::nat) < n)$
  $\langle proof \rangle$

end

3 Basics on Order-Like Relations

theory *Order-Relation-More*
  imports *Main*
begin

3.1 The upper and lower bounds operators

lemma *aboveS-subset-above*: $aboveS\ r\ a \leq above\ r\ a$
  $\langle proof \rangle$

lemma *AboveS-subset-Above*: $AboveS\ r\ A \leq Above\ r\ A$
  $\langle proof \rangle$

lemma *UnderS-disjoint*: $A\ Int\ (UnderS\ r\ A) = \{\}$
  $\langle proof \rangle$

lemma *aboveS-notIn*: $a \notin aboveS\ r\ a$
  $\langle proof \rangle$

lemma *Refl-above-in*: $[\![Refl\ r;\ a \in Field\ r]\!] \Longrightarrow a \in above\ r\ a$
  $\langle proof \rangle$

lemma *in-Above-under*: $a \in Field\ r \Longrightarrow a \in Above\ r\ (under\ r\ a)$
  $\langle proof \rangle$

lemma *in-Under-above*: $a \in Field\ r \Longrightarrow a \in Under\ r\ (above\ r\ a)$
  $\langle proof \rangle$

lemma *in-UnderS-aboveS*: $a \in Field\ r \Longrightarrow a \in UnderS\ r\ (aboveS\ r\ a)$
  $\langle proof \rangle$

lemma *UnderS-subset-Under*: $UnderS\ r\ A \leq Under\ r\ A$
  $\langle proof \rangle$

lemma *subset-Above-Under*: $B \leq \mathit{Field}\ r \Longrightarrow B \leq \mathit{Above}\ r\ (\mathit{Under}\ r\ B)$
⟨*proof*⟩

lemma *subset-Under-Above*: $B \leq \mathit{Field}\ r \Longrightarrow B \leq \mathit{Under}\ r\ (\mathit{Above}\ r\ B)$
⟨*proof*⟩

lemma *subset-AboveS-UnderS*: $B \leq \mathit{Field}\ r \Longrightarrow B \leq \mathit{AboveS}\ r\ (\mathit{UnderS}\ r\ B)$
⟨*proof*⟩

lemma *subset-UnderS-AboveS*: $B \leq \mathit{Field}\ r \Longrightarrow B \leq \mathit{UnderS}\ r\ (\mathit{AboveS}\ r\ B)$
⟨*proof*⟩

lemma *Under-Above-Galois*:
$[\![B \leq \mathit{Field}\ r;\ C \leq \mathit{Field}\ r]\!] \Longrightarrow (B \leq \mathit{Above}\ r\ C) = (C \leq \mathit{Under}\ r\ B)$
⟨*proof*⟩

lemma *UnderS-AboveS-Galois*:
$[\![B \leq \mathit{Field}\ r;\ C \leq \mathit{Field}\ r]\!] \Longrightarrow (B \leq \mathit{AboveS}\ r\ C) = (C \leq \mathit{UnderS}\ r\ B)$
⟨*proof*⟩

lemma *Refl-above-aboveS*:
assumes *REFL*: *Refl r* **and** *IN*: $a \in \mathit{Field}\ r$
shows $\mathit{above}\ r\ a = \mathit{aboveS}\ r\ a \cup \{a\}$
⟨*proof*⟩

lemma *Linear-order-under-aboveS-Field*:
assumes *LIN*: *Linear-order r* **and** *IN*: $a \in \mathit{Field}\ r$
shows $\mathit{Field}\ r = \mathit{under}\ r\ a \cup \mathit{aboveS}\ r\ a$
⟨*proof*⟩

lemma *Linear-order-underS-above-Field*:
assumes *LIN*: *Linear-order r* **and** *IN*: $a \in \mathit{Field}\ r$
shows $\mathit{Field}\ r = \mathit{underS}\ r\ a \cup \mathit{above}\ r\ a$
⟨*proof*⟩

lemma *under-empty*: $a \notin \mathit{Field}\ r \Longrightarrow \mathit{under}\ r\ a = \{\}$
⟨*proof*⟩

lemma *Under-Field*: $\mathit{Under}\ r\ A \leq \mathit{Field}\ r$
⟨*proof*⟩

lemma *UnderS-Field*: $\mathit{UnderS}\ r\ A \leq \mathit{Field}\ r$
⟨*proof*⟩

lemma *above-Field*: $\mathit{above}\ r\ a \leq \mathit{Field}\ r$
⟨*proof*⟩

lemma *aboveS-Field*: $\mathit{aboveS}\ r\ a \leq \mathit{Field}\ r$
⟨*proof*⟩

lemma *Above-Field*: *Above* r $A \leq$ *Field* r
⟨*proof*⟩

lemma *Refl-under-Under*:
assumes *REFL*: *Refl* r **and** *NE*: $A \neq \{\}$
shows *Under* r $A = (\bigcap a \in A.$ *under* r $a)$
⟨*proof*⟩

lemma *Refl-underS-UnderS*:
assumes *REFL*: *Refl* r **and** *NE*: $A \neq \{\}$
shows *UnderS* r $A = (\bigcap a \in A.$ *underS* r $a)$
⟨*proof*⟩

lemma *Refl-above-Above*:
assumes *REFL*: *Refl* r **and** *NE*: $A \neq \{\}$
shows *Above* r $A = (\bigcap a \in A.$ *above* r $a)$
⟨*proof*⟩

lemma *Refl-aboveS-AboveS*:
assumes *REFL*: *Refl* r **and** *NE*: $A \neq \{\}$
shows *AboveS* r $A = (\bigcap a \in A.$ *aboveS* r $a)$
⟨*proof*⟩

lemma *under-Under-singl*: *under* r $a =$ *Under* r $\{a\}$
⟨*proof*⟩

lemma *underS-UnderS-singl*: *underS* r $a =$ *UnderS* r $\{a\}$
⟨*proof*⟩

lemma *above-Above-singl*: *above* r $a =$ *Above* r $\{a\}$
⟨*proof*⟩

lemma *aboveS-AboveS-singl*: *aboveS* r $a =$ *AboveS* r $\{a\}$
⟨*proof*⟩

lemma *Under-decr*: $A \leq B \Longrightarrow$ *Under* r $B \leq$ *Under* r A
⟨*proof*⟩

lemma *UnderS-decr*: $A \leq B \Longrightarrow$ *UnderS* r $B \leq$ *UnderS* r A
⟨*proof*⟩

lemma *Above-decr*: $A \leq B \Longrightarrow$ *Above* r $B \leq$ *Above* r A
⟨*proof*⟩

lemma *AboveS-decr*: $A \leq B \Longrightarrow$ *AboveS* r $B \leq$ *AboveS* r A
⟨*proof*⟩

lemma *under-incl-iff*:

assumes *TRANS*: *trans r* **and** *REFL*: *Refl r* **and** *IN*: $a \in$ *Field r*
shows (*under r a* $\leq$ *under r b*) = ((a,b) $\in r$)
⟨*proof*⟩

lemma *above-decr*:
assumes *TRANS*: *trans r* **and** *REL*: (a,b) $\in r$
shows *above r b* $\leq$ *above r a*
⟨*proof*⟩

lemma *aboveS-decr*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
REL: (a,b) $\in r$
shows *aboveS r b* $\leq$ *aboveS r a*
⟨*proof*⟩

lemma *under-trans*:
assumes *TRANS*: *trans r* **and**
IN1: $a \in$ *under r b* **and** *IN2*: $b \in$ *under r c*
shows $a \in$ *under r c*
⟨*proof*⟩

lemma *under-underS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in$ *under r b* **and** *IN2*: $b \in$ *underS r c*
shows $a \in$ *underS r c*
⟨*proof*⟩

lemma *underS-under-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in$ *underS r b* **and** *IN2*: $b \in$ *under r c*
shows $a \in$ *underS r c*
⟨*proof*⟩

lemma *underS-underS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in$ *underS r b* **and** *IN2*: $b \in$ *underS r c*
shows $a \in$ *underS r c*
⟨*proof*⟩

lemma *above-trans*:
assumes *TRANS*: *trans r* **and**
IN1: $b \in$ *above r a* **and** *IN2*: $c \in$ *above r b*
shows $c \in$ *above r a*
⟨*proof*⟩

lemma *above-aboveS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $b \in$ *above r a* **and** *IN2*: $c \in$ *aboveS r b*
shows $c \in$ *aboveS r a*

⟨*proof*⟩

lemma *aboveS-above-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $b \in$ *aboveS r a* **and** *IN2*: $c \in$ *above r b*
  shows $c \in$ *aboveS r a*
⟨*proof*⟩

lemma *aboveS-aboveS-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $b \in$ *aboveS r a* **and** *IN2*: $c \in$ *aboveS r b*
  shows $c \in$ *aboveS r a*
⟨*proof*⟩

lemma *under-Under-trans*:
  assumes *TRANS*: *trans r* **and**
    IN1: $a \in$ *under r b* **and** *IN2*: $b \in$ *Under r C*
  shows $a \in$ *Under r C*
⟨*proof*⟩

lemma *underS-Under-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $a \in$ *underS r b* **and** *IN2*: $b \in$ *Under r C*
  shows $a \in$ *UnderS r C*
⟨*proof*⟩

lemma *underS-UnderS-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $a \in$ *underS r b* **and** *IN2*: $b \in$ *UnderS r C*
  shows $a \in$ *UnderS r C*
⟨*proof*⟩

lemma *above-Above-trans*:
  assumes *TRANS*: *trans r* **and**
    IN1: $a \in$ *above r b* **and** *IN2*: $b \in$ *Above r C*
  shows $a \in$ *Above r C*
⟨*proof*⟩

lemma *aboveS-Above-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $a \in$ *aboveS r b* **and** *IN2*: $b \in$ *Above r C*
  shows $a \in$ *AboveS r C*
⟨*proof*⟩

lemma *above-AboveS-trans*:
  assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
    IN1: $a \in$ *above r b* **and** *IN2*: $b \in$ *AboveS r C*
  shows $a \in$ *AboveS r C*
⟨*proof*⟩

lemma *aboveS-AboveS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in$ *aboveS r b* **and** *IN2*: $b \in$ *AboveS r C*
shows $a \in$ *AboveS r C*
⟨*proof*⟩

lemma *under-UnderS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in$ *under r b* **and** *IN2*: $b \in$ *UnderS r C*
shows $a \in$ *UnderS r C*
⟨*proof*⟩

3.2 Properties depending on more than one relation

lemma *under-incr2*:
$r \leq r' \Longrightarrow$ *under r a* $\leq$ *under r' a*
⟨*proof*⟩

lemma *underS-incr2*:
$r \leq r' \Longrightarrow$ *underS r a* $\leq$ *underS r' a*
⟨*proof*⟩

lemma *above-incr2*:
$r \leq r' \Longrightarrow$ *above r a* $\leq$ *above r' a*
⟨*proof*⟩

lemma *aboveS-incr2*:
$r \leq r' \Longrightarrow$ *aboveS r a* $\leq$ *aboveS r' a*
⟨*proof*⟩

end

4 More on Well-Founded Relations

theory *Wellfounded-More*
imports *Main Order-Relation-More*
begin

4.1 Well-founded recursion via genuine fixpoints

lemma *adm-wf-unique-fixpoint*:
fixes $r :: ('a * 'a)$ *set* **and**
$H :: ('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$ **and**
$f :: 'a \Rightarrow 'b$ **and** $g :: 'a \Rightarrow 'b$

assumes *WF*: *wf r* **and** *ADM*: *adm-wf r H* **and** *fFP*: $f = H\,f$ **and** *gFP*: $g = H\,g$
shows $f = g$
⟨*proof*⟩

lemma *wfrec-unique-fixpoint*:
fixes $r :: ('a * 'a)$ *set* **and**
$H :: ('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$ **and**
$f :: 'a \Rightarrow 'b$
assumes *WF*: *wf r* **and** *ADM*: *adm-wf r H* **and**
fp: $f = H\,f$
shows $f = wfrec\ r\ H$
⟨*proof*⟩

end

5 Well-Order Relations

theory *Wellorder-Relation*
imports *Wellfounded-More*
begin

context *wo-rel*
begin

5.1 Auxiliaries

lemma *PREORD*: *Preorder r*
⟨*proof*⟩

lemma *PARORD*: *Partial-order r*
⟨*proof*⟩

lemma *cases-Total2*:
$\bigwedge phi\ a\ b.\ [\![\{a,b\} \leq Field\ r;\ ((a,b) \in r - Id \Longrightarrow phi\ a\ b);$
$((b,a) \in r - Id \Longrightarrow phi\ a\ b);\ (a = b \Longrightarrow phi\ a\ b)]\!]$
$\Longrightarrow phi\ a\ b$
⟨*proof*⟩

5.2 Well-founded induction and recursion adapted to non-strict well-order relations

lemma *worec-unique-fixpoint*:
assumes *ADM*: *adm-wo H* **and** *fp*: $f = H\,f$
shows $f = worec\ H$
⟨*proof*⟩

5.2.1 Properties of max2

lemma *max2-iff*:

assumes $a \in \mathit{Field}\ r$ **and** $b \in \mathit{Field}\ r$
shows $((\mathit{max2}\ a\ b,\ c) \in r) = ((a,c) \in r \wedge (b,c) \in r)$
$\langle \mathit{proof} \rangle$

5.2.2 Properties of minim

lemma *minim-Under*:
$[\![B \leq \mathit{Field}\ r;\ B \neq \{\}]\!] \Longrightarrow \mathit{minim}\ B \in \mathit{Under}\ B$
$\langle \mathit{proof} \rangle$

lemma *equals-minim-Under*:
$[\![B \leq \mathit{Field}\ r;\ a \in B;\ a \in \mathit{Under}\ B]\!]$
$\Longrightarrow a = \mathit{minim}\ B$
$\langle \mathit{proof} \rangle$

lemma *minim-iff-In-Under*:
assumes *SUB*: $B \leq \mathit{Field}\ r$ **and** *NE*: $B \neq \{\}$
shows $(a = \mathit{minim}\ B) = (a \in B \wedge a \in \mathit{Under}\ B)$
$\langle \mathit{proof} \rangle$

lemma *minim-Under-under*:
assumes *NE*: $A \neq \{\}$ **and** *SUB*: $A \leq \mathit{Field}\ r$
shows $\mathit{Under}\ A = \mathit{under}\ (\mathit{minim}\ A)$
$\langle \mathit{proof} \rangle$

lemma *minim-UnderS-underS*:
assumes *NE*: $A \neq \{\}$ **and** *SUB*: $A \leq \mathit{Field}\ r$
shows $\mathit{UnderS}\ A = \mathit{underS}\ (\mathit{minim}\ A)$
$\langle \mathit{proof} \rangle$

5.2.3 Properties of supr

lemma *supr-Above*:
assumes $\mathit{Above}\ B \neq \{\}$
shows $\mathit{supr}\ B \in \mathit{Above}\ B$
$\langle \mathit{proof} \rangle$

lemma *supr-greater*:
assumes $\mathit{Above}\ B \neq \{\}\ b \in B$
shows $(b,\ \mathit{supr}\ B) \in r$
$\langle \mathit{proof} \rangle$

lemma *supr-least-Above*:
assumes $a \in \mathit{Above}\ B$
shows $(\mathit{supr}\ B,\ a) \in r$
$\langle \mathit{proof} \rangle$

lemma *supr-least*:
$[\![B \leq \mathit{Field}\ r;\ a \in \mathit{Field}\ r;\ (\bigwedge b.\ b \in B \Longrightarrow (b,a) \in r)]\!]$
$\Longrightarrow (\mathit{supr}\ B,\ a) \in r$

$\langle proof \rangle$

lemma *equals-supr-Above*:
assumes $a \in \mathit{Above}\ B\ \bigwedge a'.\ a' \in \mathit{Above}\ B \Longrightarrow (a,a') \in r$
shows $a = \mathit{supr}\ B$
$\langle proof \rangle$

lemma *equals-supr*:
assumes SUB: $B \leq \mathit{Field}\ r$ **and** IN: $a \in \mathit{Field}\ r$ **and**
ABV: $\bigwedge b.\ b \in B \Longrightarrow (b,a) \in r$ **and**
MINIM: $\bigwedge a'.\ [\![\ a' \in \mathit{Field}\ r;\ \bigwedge b.\ b \in B \Longrightarrow (b,a') \in r]\!] \Longrightarrow (a,a') \in r$
shows $a = \mathit{supr}\ B$
$\langle proof \rangle$

lemma *supr-inField*:
assumes $\mathit{Above}\ B \neq \{\}$
shows $\mathit{supr}\ B \in \mathit{Field}\ r$
$\langle proof \rangle$

lemma *supr-above-Above*:
assumes SUB: $B \leq \mathit{Field}\ r$ **and** ABOVE: $\mathit{Above}\ B \neq \{\}$
shows $\mathit{Above}\ B = \mathit{above}\ (\mathit{supr}\ B)$
$\langle proof \rangle$

lemma *supr-under*:
assumes $a \in \mathit{Field}\ r$
shows $a = \mathit{supr}\ (\mathit{under}\ a)$
$\langle proof \rangle$

5.2.4 Properties of successor

lemma *suc-least*:
$[\![B \leq \mathit{Field}\ r;\ a \in \mathit{Field}\ r;\ (\bigwedge b.\ b \in B \Longrightarrow a \neq b \wedge (b,a) \in r)]\!]$
$\Longrightarrow (\mathit{suc}\ B,\ a) \in r$
$\langle proof \rangle$

lemma *equals-suc*:
assumes SUB: $B \leq \mathit{Field}\ r$ **and** IN: $a \in \mathit{Field}\ r$ **and**
ABVS: $\bigwedge b.\ b \in B \Longrightarrow a \neq b \wedge (b,a) \in r$ **and**
MINIM: $\bigwedge a'.\ [\![a' \in \mathit{Field}\ r;\ \bigwedge b.\ b \in B \Longrightarrow a' \neq b \wedge (b,a') \in r]\!] \Longrightarrow (a,a') \in$
r
shows $a = \mathit{suc}\ B$
$\langle proof \rangle$

lemma *suc-above-AboveS*:
assumes SUB: $B \leq \mathit{Field}\ r$ **and**
ABOVE: $\mathit{AboveS}\ B \neq \{\}$
shows $\mathit{AboveS}\ B = \mathit{above}\ (\mathit{suc}\ B)$
$\langle proof \rangle$

lemma *suc-singl-pred*:
  assumes *IN*: $a \in \mathit{Field}\ r$ **and** *ABOVE-NE*: $\mathit{aboveS}\ a \neq \{\}$ **and**
    REL: $(a',\mathit{suc}\ \{a\}) \in r$ **and** *DIFF*: $a' \neq \mathit{suc}\ \{a\}$
  shows $a' = a \vee (a',a) \in r$
⟨*proof*⟩

lemma *under-underS-suc*:
  assumes *IN*: $a \in \mathit{Field}\ r$ **and** *ABV*: $\mathit{aboveS}\ a \neq \{\}$
  shows $\mathit{underS}\ (\mathit{suc}\ \{a\}) = \mathit{under}\ a$
⟨*proof*⟩

5.2.5 Properties of order filters

lemma *ofilter-Under*[*simp*]:
  assumes $A \leq \mathit{Field}\ r$
  shows $\mathit{ofilter}(\mathit{Under}\ A)$
  ⟨*proof*⟩

lemma *ofilter-UnderS*[*simp*]:
  assumes $A \leq \mathit{Field}\ r$
  shows $\mathit{ofilter}(\mathit{UnderS}\ A)$
  ⟨*proof*⟩

lemma *ofilter-Int*[*simp*]: $[\![\mathit{ofilter}\ A;\ \mathit{ofilter}\ B]\!] \Longrightarrow \mathit{ofilter}(A\ \mathit{Int}\ B)$
  ⟨*proof*⟩

lemma *ofilter-Un*[*simp*]: $[\![\mathit{ofilter}\ A;\ \mathit{ofilter}\ B]\!] \Longrightarrow \mathit{ofilter}(A \cup B)$
  ⟨*proof*⟩

lemma *ofilter-INTER*:
  $[\![I \neq \{\};\ \bigwedge i.\ i \in I \Longrightarrow \mathit{ofilter}(A\ i)]\!] \Longrightarrow \mathit{ofilter}\ (\bigcap i \in I.\ A\ i)$
  ⟨*proof*⟩

lemma *ofilter-Inter*:
  $[\![S \neq \{\};\ \bigwedge A.\ A \in S \Longrightarrow \mathit{ofilter}\ A]\!] \Longrightarrow \mathit{ofilter}\ (\bigcap S)$
  ⟨*proof*⟩

lemma *ofilter-Union*:
  $(\bigwedge A.\ A \in S \Longrightarrow \mathit{ofilter}\ A) \Longrightarrow \mathit{ofilter}\ (\bigcup S)$
  ⟨*proof*⟩

lemma *ofilter-under-Union*:
  $\mathit{ofilter}\ A \Longrightarrow A = \bigcup\{\mathit{under}\ a|\ a.\ a \in A\}$
  ⟨*proof*⟩

5.2.6 Other properties

lemma *Trans-Under-regressive*:
  assumes *NE*: $A \neq \{\}$ **and** *SUB*: $A \leq \mathit{Field}\ r$

shows $Under(Under\ A) \leq Under\ A$
⟨*proof*⟩

lemma *ofilter-suc-Field*:
assumes *OF*: *ofilter A* **and** *NE*: $A \neq Field\ r$
shows *ofilter* $(A \cup \{suc\ A\})$
⟨*proof*⟩

declare
minim-in[*simp*]
minim-inField[*simp*]
minim-least[*simp*]
under-ofilter[*simp*]
underS-ofilter[*simp*]
Field-ofilter[*simp*]

end

abbreviation *worec* ≡ *wo-rel.worec*
abbreviation *adm-wo* ≡ *wo-rel.adm-wo*
abbreviation *isMinim* ≡ *wo-rel.isMinim*
abbreviation *minim* ≡ *wo-rel.minim*
abbreviation *max2* ≡ *wo-rel.max2*
abbreviation *supr* ≡ *wo-rel.supr*
abbreviation *suc* ≡ *wo-rel.suc*

end

6 Well-Order Embeddings

theory *Wellorder-Embedding*
imports *Fun-More Wellorder-Relation*
begin

6.1 Auxiliaries

lemma *UNION-bij-betw-ofilter*:
assumes *WELL*: *Well-order r* **and**
OF: $\bigwedge i.\ i \in I \Longrightarrow$ *ofilter r* $(A\ i)$ **and**
BIJ: $\bigwedge i.\ i \in I \Longrightarrow$ *bij-betw f* $(A\ i)\ (A'\ i)$
shows *bij-betw f* $(\bigcup i \in I.\ A\ i)\ (\bigcup i \in I.\ A'\ i)$
⟨*proof*⟩

6.2 (Well-order) embeddings, strict embeddings, isomorphisms and order-compatible functions

lemma *embed-halfcong*:
assumes $\bigwedge a.\ a \in Field\ r \Longrightarrow f\ a = g\ a$ **and** *embed* $r\ r'\ f$

shows $embed\ r\ r'\ g$
$\langle proof \rangle$

lemma *embed-cong*[*fundef-cong*]:
assumes $\bigwedge\ a.\ a \in Field\ r \Longrightarrow f\ a = g\ a$
shows $embed\ r\ r'\ f = embed\ r\ r'\ g$
$\langle proof \rangle$

lemma *embedS-cong*[*fundef-cong*]:
assumes $\bigwedge\ a.\ a \in Field\ r \Longrightarrow f\ a = g\ a$
shows $embedS\ r\ r'\ f = embedS\ r\ r'\ g$
$\langle proof \rangle$

lemma *iso-cong*[*fundef-cong*]:
assumes $\bigwedge\ a.\ a \in Field\ r \Longrightarrow f\ a = g\ a$
shows $iso\ r\ r'\ f = iso\ r\ r'\ g$
$\langle proof \rangle$

lemma *id-compat*: $compat\ r\ r\ id$
$\langle proof \rangle$

lemma *comp-compat*:
$[\![compat\ r\ r'\ f;\ compat\ r'\ r''\ f']\!] \Longrightarrow compat\ r\ r''\ (f'\ o\ f)$
$\langle proof \rangle$

corollary *one-set-greater*:
$(\exists f{::}'a \Rightarrow 'a'.\ f\ `\ A \leq A' \wedge inj\text{-}on\ f\ A) \vee (\exists g{::}'a' \Rightarrow 'a.\ g\ `\ A' \leq A \wedge inj\text{-}on\ g\ A')$
$\langle proof \rangle$

corollary *one-type-greater*:
$(\exists f{::}'a \Rightarrow 'a'.\ inj\ f) \vee (\exists g{::}'a' \Rightarrow 'a.\ inj\ g)$
$\langle proof \rangle$

6.3 Uniqueness of embeddings

lemma *comp-embedS*:
assumes $WELL$: $Well\text{-}order\ r$ **and** $WELL'$: $Well\text{-}order\ r'$ **and** $WELL''$: $Well\text{-}order\ r''$
and EMB: $embedS\ r\ r'\ f$ **and** EMB': $embedS\ r'\ r''\ f'$
shows $embedS\ r\ r''\ (f'\ o\ f)$
$\langle proof \rangle$

lemma *iso-iff4*:
assumes $WELL$: $Well\text{-}order\ r$ **and** $WELL'$: $Well\text{-}order\ r'$
shows $iso\ r\ r'\ f = (embed\ r\ r'\ f \wedge embed\ r'\ r\ (inv\text{-}into\ (Field\ r)\ f))$
$\langle proof \rangle$

lemma *embed-embedS-iso*:

$embed\ r\ r'\ f = (embedS\ r\ r'\ f \vee iso\ r\ r'\ f)$
⟨*proof*⟩

lemma *not-embedS-iso*:
$\neg\ (embedS\ r\ r'\ f \wedge iso\ r\ r'\ f)$
⟨*proof*⟩

lemma *embed-embedS-iff-not-iso*:
assumes $embed\ r\ r'\ f$
shows $embedS\ r\ r'\ f = (\neg\ iso\ r\ r'\ f)$
⟨*proof*⟩

lemma *iso-inv-into*:
assumes *WELL*: *Well-order* r **and** *ISO*: $iso\ r\ r'\ f$
shows $iso\ r'\ r\ (inv\text{-}into\ (Field\ r)\ f)$
⟨*proof*⟩

lemma *embedS-or-iso*:
assumes *WELL*: *Well-order* r **and** *WELL'*: *Well-order* r'
shows $(\exists\, g.\ embedS\ r\ r'\ g) \vee (\exists\, h.\ embedS\ r'\ r\ h) \vee (\exists\, f.\ iso\ r\ r'\ f)$
⟨*proof*⟩

end

7 Order Union

theory *Order-Union*
imports *Main*
begin

definition $Osum :: 'a\ rel \Rightarrow 'a\ rel \Rightarrow 'a\ rel$ (**infix** *Osum 60*) **where**
$r\ Osum\ r' = r \cup r' \cup \{(a,\ a').\ a \in Field\ r \wedge a' \in Field\ r'\}$

notation *Osum* (**infix** $\cup o$ *60*)

lemma *Field-Osum*: $Field\ (r \cup o\ r') = Field\ r \cup Field\ r'$
⟨*proof*⟩

lemma *Osum-wf*:
assumes *FLD*: $Field\ r\ Int\ Field\ r' = \{\}$ **and**
WF: $wf\ r$ **and** *WF'*: $wf\ r'$
shows $wf\ (r\ Osum\ r')$
⟨*proof*⟩

lemma *Osum-Refl*:
assumes *FLD*: $Field\ r\ Int\ Field\ r' = \{\}$ **and**
REFL: $Refl\ r$ **and** *REFL'*: $Refl\ r'$
shows $Refl\ (r\ Osum\ r')$
⟨*proof*⟩

lemma *Osum-trans*:
assumes *FLD*: *Field r Int Field r′* = {} **and**
TRANS: *trans r* **and** *TRANS′*: *trans r′*
shows *trans* (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-Preorder*:
⟦*Field r Int Field r′* = {}; *Preorder r*; *Preorder r′*⟧ $\Longrightarrow$ *Preorder* (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-antisym*:
assumes *FLD*: *Field r Int Field r′* = {} **and**
AN: *antisym r* **and** *AN′*: *antisym r′*
shows *antisym* (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-Partial-order*:
⟦*Field r Int Field r′* = {}; *Partial-order r*; *Partial-order r′*⟧ $\Longrightarrow$
Partial-order (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-Total*:
assumes *FLD*: *Field r Int Field r′* = {} **and**
TOT: *Total r* **and** *TOT′*: *Total r′*
shows *Total* (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-Linear-order*:
⟦*Field r Int Field r′* = {}; *Linear-order r*; *Linear-order r′*⟧ $\Longrightarrow$ *Linear-order* (*r Osum r′*)
⟨*proof*⟩

lemma *Osum-minus-Id1*:
assumes $r \leq Id$
shows $(r\ Osum\ r') - Id \leq (r' - Id) \cup (Field\ r \times Field\ r')$
⟨*proof*⟩

lemma *Osum-minus-Id2*:
assumes $r' \leq Id$
shows $(r\ Osum\ r') - Id \leq (r - Id) \cup (Field\ r \times Field\ r')$
⟨*proof*⟩

lemma *Osum-minus-Id*:
assumes *TOT*: *Total r* **and** *TOT′*: *Total r′* **and**
NID: $\neg\ (r \leq Id)$ **and** *NID′*: $\neg\ (r' \leq Id)$
shows $(r\ Osum\ r') - Id \leq (r - Id)\ Osum\ (r' - Id)$
⟨*proof*⟩

lemma *wf-Int-Times*:
assumes A *Int* $B = \{\}$
shows $wf(A \times B)$
⟨*proof*⟩

lemma *Osum-wf-Id*:
assumes *TOT*: *Total* r **and** TOT': *Total* r' **and**
FLD: *Field* r *Int* *Field* $r' = \{\}$ **and**
WF: $wf(r - Id)$ **and** WF': $wf(r' - Id)$
shows $wf\ ((r\ Osum\ r') - Id)$
⟨*proof*⟩

lemma *Osum-Well-order*:
assumes *FLD*: *Field* r *Int* *Field* $r' = \{\}$ **and**
WELL: *Well-order* r **and** $WELL'$: *Well-order* r'
shows *Well-order* $(r\ Osum\ r')$
⟨*proof*⟩

end

8 Constructions on Wellorders

theory *Wellorder-Constructions*
imports
Wellorder-Embedding Order-Union
begin

unbundle *cardinal-syntax*

declare
ordLeq-Well-order-simp[*simp*]
not-ordLeq-iff-ordLess[*simp*]
not-ordLess-iff-ordLeq[*simp*]
Func-empty[*simp*]
Func-is-emp[*simp*]

8.1 Order filters versus restrictions and embeddings

lemma *ofilter-subset-iso*:
assumes *WELL*: *Well-order* r **and**
OFA: *ofilter* r A **and** *OFB*: *ofilter* r B
shows $(A = B) = iso\ (Restr\ r\ A)\ (Restr\ r\ B)\ id$
⟨*proof*⟩

8.2 Ordering the well-orders by existence of embeddings

corollary *ordLeq-refl-on*: *refl-on* $\{r.\ Well\text{-}order\ r\}$ *ordLeq*
⟨*proof*⟩

corollary *ordLeq-trans*: *trans ordLeq*
⟨*proof*⟩

corollary *ordLeq-preorder-on*: *preorder-on* $\{r.$ *Well-order* $r\}$ *ordLeq*
⟨*proof*⟩

corollary *ordIso-subset*: *ordIso* $\subseteq \{r.$ *Well-order* $r\} \times \{r.$ *Well-order* $r\}$
⟨*proof*⟩

corollary *ordIso-refl-on*: *refl-on* $\{r.$ *Well-order* $r\}$ *ordIso*
⟨*proof*⟩

corollary *ordIso-trans*: *trans ordIso*
⟨*proof*⟩

corollary *ordIso-sym*: *sym ordIso*
⟨*proof*⟩

corollary *ordIso-equiv*: *equiv* $\{r.$ *Well-order* $r\}$ *ordIso*
⟨*proof*⟩

lemma *ordLess-Well-order-simp*[*simp*]:
assumes $r <o\ r'$
shows *Well-order* $r \wedge$ *Well-order* r'
⟨*proof*⟩

lemma *ordIso-Well-order-simp*[*simp*]:
assumes $r =o\ r'$
shows *Well-order* $r \wedge$ *Well-order* r'
⟨*proof*⟩

lemma *ordLess-irrefl*: *irrefl ordLess*
⟨*proof*⟩

lemma *ordLess-or-ordIso*:
assumes *WELL*: *Well-order* r **and** *WELL'*: *Well-order* r'
shows $r <o\ r' \vee r' <o\ r \vee r =o\ r'$
⟨*proof*⟩

corollary *ordLeq-ordLess-Un-ordIso*:
ordLeq = *ordLess* $\cup$ *ordIso*
⟨*proof*⟩

lemma *ordIso-or-ordLess*:
assumes *WELL*: *Well-order* r **and** *WELL'*: *Well-order* r'
shows $r =o\ r' \vee r <o\ r' \vee r' <o\ r$
⟨*proof*⟩

lemmas *ord-trans* = *ordIso-transitive ordLeq-transitive ordLess-transitive*

ordIso-ordLeq-trans ordLeq-ordIso-trans
ordIso-ordLess-trans ordLess-ordIso-trans
ordLess-ordLeq-trans ordLeq-ordLess-trans

lemma *ofilter-ordLeq*:
assumes *Well-order r* **and** *ofilter r A*
shows *Restr r A* $\leq o$ *r*
⟨*proof*⟩

corollary *under-Restr-ordLeq*:
Well-order r $\Longrightarrow$ *Restr r* (*under r a*) $\leq o$ *r*
⟨*proof*⟩

8.3 Copy via direct images

lemma *Id-dir-image*: *dir-image Id f* $\leq$ *Id*
⟨*proof*⟩

lemma *Un-dir-image*:
dir-image (*r1* $\cup$ *r2*) *f* = (*dir-image r1 f*) $\cup$ (*dir-image r2 f*)
⟨*proof*⟩

lemma *Int-dir-image*:
assumes *inj-on f* (*Field r1* $\cup$ *Field r2*)
shows *dir-image* (*r1 Int r2*) *f* = (*dir-image r1 f*) *Int* (*dir-image r2 f*)
⟨*proof*⟩

lemma *Osum-embed*:
assumes *FLD*: *Field r Int Field* r' = {} **and**
WELL: *Well-order r* **and** *WELL'*: *Well-order* r'
shows *embed r* (*r Osum* r') *id*
⟨*proof*⟩

corollary *Osum-ordLeq*:
assumes *FLD*: *Field r Int Field* r' = {} **and**
WELL: *Well-order r* **and** *WELL'*: *Well-order* r'
shows *r* $\leq o$ *r Osum* r'
⟨*proof*⟩

lemma *Well-order-embed-copy*:
assumes *WELL*: *well-order-on A r* **and**
INJ: *inj-on f A* **and** *SUB*: *f ‘ A* $\leq$ *B*
shows $\exists\, r'$. *well-order-on B* $r' \wedge r \leq o\ r'$
⟨*proof*⟩

8.4 The maxim among a finite set of ordinals

The correct phrasing would be "a maxim of ...", as $\leq o$ is only a preorder.

definition *isOmax* :: $'a$ *rel set* $\Rightarrow$ $'a$ *rel* $\Rightarrow$ *bool*
  where
    isOmax $R\ r \equiv r \in R \wedge (\forall r' \in R.\ r' \leq o\ r)$

definition *omax* :: $'a$ *rel set* $\Rightarrow$ $'a$ *rel*
  where
    omax R == *SOME* r. *isOmax* $R\ r$

lemma *exists-isOmax*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall\ r \in R.$ *Well-order* r
  shows $\exists\ r.$ *isOmax* $R\ r$
$\langle proof \rangle$

lemma *omax-isOmax*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall\ r \in R.$ *Well-order* r
  shows *isOmax* R (*omax* R)
  $\langle proof \rangle$

lemma *omax-in*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall\ r \in R.$ *Well-order* r
  shows *omax* $R \in R$
  $\langle proof \rangle$

lemma *Well-order-omax*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall r{\in}R.$ *Well-order* r
  shows *Well-order* (*omax* R)
  $\langle proof \rangle$

lemma *omax-maxim*:
  assumes *finite* R **and** $\forall\ r \in R.$ *Well-order* r **and** $r \in R$
  shows $r \leq o$ *omax* R
  $\langle proof \rangle$

lemma *omax-ordLeq*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall\ r \in R.\ r \leq o\ p$
  shows *omax* $R \leq o\ p$
  $\langle proof \rangle$

lemma *omax-ordLess*:
  assumes *finite* R **and** $R \neq \{\}$ **and** $\forall\ r \in R.\ r < o\ p$
  shows *omax* $R < o\ p$
  $\langle proof \rangle$

lemma *omax-ordLeq-elim*:
  assumes *finite* R **and** $\forall\ r \in R.$ *Well-order* r
    and *omax* $R \leq o\ p$ **and** $r \in R$

shows $r \leq o\ p$
⟨*proof*⟩

lemma *omax-ordLess-elim*:
assumes *finite* R **and** $\forall\ r \in R.$ *Well-order* r
and *omax* $R <o\ p$ **and** $r \in R$
shows $r <o\ p$
⟨*proof*⟩

lemma *ordLeq-omax*:
assumes *finite* R **and** $\forall\ r \in R.$ *Well-order* r
and $r \in R$ **and** $p \leq o\ r$
shows $p \leq o$ *omax* R
⟨*proof*⟩

lemma *ordLess-omax*:
assumes *finite* R **and** $\forall\ r \in R.$ *Well-order* r
and $r \in R$ **and** $p <o\ r$
shows $p <o$ *omax* R
⟨*proof*⟩

lemma *omax-ordLeq-mono*:
assumes P: *finite* P **and** R: *finite* R
and *NE-P*: $P \neq \{\}$ **and** *Well-R*: $\forall\ r \in R.$ *Well-order* r
and *LEQ*: $\forall\ p \in P.\ \exists\ r \in R.\ p \leq o\ r$
shows *omax* $P \leq o$ *omax* R
⟨*proof*⟩

lemma *omax-ordLess-mono*:
assumes P: *finite* P **and** R: *finite* R
and *NE-P*: $P \neq \{\}$ **and** *Well-R*: $\forall\ r \in R.$ *Well-order* r
and *LEQ*: $\forall\ p \in P.\ \exists\ r \in R.\ p <o\ r$
shows *omax* $P <o$ *omax* R
⟨*proof*⟩

8.5 Limit and succesor ordinals

lemma *embed-underS2*:
assumes r: *Well-order* r **and** g: *embed* $r\ s\ g$ **and** a: $a \in$ *Field* r
shows g ‘ *underS* $r\ a =$ *underS* $s\ (g\ a)$
⟨*proof*⟩

lemma *bij-betw-insert*:
assumes $b \notin A$ **and** $f\ b \notin A'$ **and** *bij-betw* $f\ A\ A'$
shows *bij-betw* f (*insert* $b\ A$) (*insert* $(f\ b)\ A'$)
⟨*proof*⟩

context *wo-rel*
begin

lemma *underS-induct*:
assumes $\bigwedge a.\ (\bigwedge\ a1.\ a1 \in underS\ a \Longrightarrow P\ a1) \Longrightarrow P\ a$
shows $P\ a$
⟨*proof*⟩

lemma *suc-underS*:
assumes B: $B \subseteq Field\ r$ **and** A: $AboveS\ B \neq \{\}$ **and** b: $b \in B$
shows $b \in underS\ (suc\ B)$
⟨*proof*⟩

lemma *underS-supr*:
assumes bA: $b \in underS\ (supr\ A)$ **and** A: $A \subseteq Field\ r$
shows $\exists\ a \in A.\ b \in underS\ a$
⟨*proof*⟩

lemma *underS-suc*:
assumes bA: $b \in underS\ (suc\ A)$ **and** A: $A \subseteq Field\ r$
shows $\exists\ a \in A.\ b \in under\ a$
⟨*proof*⟩

lemma (**in** *wo-rel*) *in-underS-supr*:
assumes $j \in underS\ i$ **and** $i \in A$ **and** $A \subseteq Field\ r$ **and** $Above\ A \neq \{\}$
shows $j \in underS\ (supr\ A)$
⟨*proof*⟩

lemma *inj-on-Field*:
assumes A: $A \subseteq Field\ r$ **and** f: $\bigwedge\ a\ b.\ [\![a \in A;\ b \in A;\ a \in underS\ b]\!] \Longrightarrow f\ a \neq f\ b$
shows $inj\text{-}on\ f\ A$
⟨*proof*⟩

lemma *ofilter-init-seg-of*:
assumes *ofilter F*
shows *Restr r F initial-segment-of r*
⟨*proof*⟩

lemma *underS-init-seg-of-Collect*:
assumes $\bigwedge j1\ j2.\ [\![j2 \in underS\ i;\ (j1,\ j2) \in r]\!] \Longrightarrow R\ j1\ initial\text{-}segment\text{-}of\ R\ j2$
shows $\{R\ j\ |j.\ j \in underS\ i\} \in Chains\ init\text{-}seg\text{-}of$
⟨*proof*⟩

lemma (**in** *wo-rel*) *Field-init-seg-of-Collect*:
assumes $\bigwedge j1\ j2.\ [\![j2 \in Field\ r;\ (j1,\ j2) \in r]\!] \Longrightarrow R\ j1\ initial\text{-}segment\text{-}of\ R\ j2$
shows $\{R\ j\ |j.\ j \in Field\ r\} \in Chains\ init\text{-}seg\text{-}of$
⟨*proof*⟩

8.5.1 Successor and limit elements of an ordinal

definition *succ* $i \equiv$ *suc* $\{i\}$

definition *isSucc* $i \equiv \exists\ j.$ *aboveS* $j \neq \{\} \wedge i =$ *succ* j

definition *zero* $=$ *minim* (*Field* r)

definition *isLim* $i \equiv \neg$ *isSucc* i

lemma *zero-smallest*[*simp*]:
assumes $j \in$ *Field* r **shows** (*zero*, j) $\in r$
⟨*proof*⟩

lemma *zero-in-Field*: **assumes** *Field* $r \neq \{\}$ **shows** *zero* $\in$ *Field* r
⟨*proof*⟩

lemma *leq-zero-imp*[*simp*]:
(x, *zero*) $\in r \Longrightarrow x =$ *zero*
⟨*proof*⟩

lemma *leq-zero*[*simp*]:
assumes *Field* $r \neq \{\}$ **shows** (x, *zero*) $\in r \longleftrightarrow x =$ *zero*
⟨*proof*⟩

lemma *under-zero*[*simp*]:
assumes *Field* $r \neq \{\}$ **shows** *under zero* $= \{$*zero*$\}$
⟨*proof*⟩

lemma *underS-zero*[*simp*,*intro*]: *underS zero* $= \{\}$
⟨*proof*⟩

lemma *isSucc-succ*: *aboveS* $i \neq \{\} \Longrightarrow$ *isSucc* (*succ* i)
⟨*proof*⟩

lemma *succ-in-diff*:
assumes *aboveS* $i \neq \{\}$ **shows** (i,*succ* i) $\in r \wedge$ *succ* $i \neq i$
⟨*proof*⟩

lemmas *succ-in*[*simp*] = *succ-in-diff*[*THEN conjunct1*]
lemmas *succ-diff*[*simp*] = *succ-in-diff*[*THEN conjunct2*]

lemma *succ-in-Field*[*simp*]:
assumes *aboveS* $i \neq \{\}$ **shows** *succ* $i \in$ *Field* r
⟨*proof*⟩

lemma *succ-not-in*:
assumes *aboveS* $i \neq \{\}$ **shows** (*succ* i, i) $\notin r$
⟨*proof*⟩

lemma *not-isSucc-zero*: $\neg$ *isSucc zero*
⟨*proof*⟩

lemma *isLim-zero*[*simp*]: *isLim zero*
⟨*proof*⟩

lemma *succ-smallest*:
assumes $(i,j) \in r$ **and** $i \neq j$
shows $(succ\ i,\ j) \in r$
⟨*proof*⟩

lemma *isLim-supr*:
assumes *f*: $i \in Field\ r$ **and** *l*: *isLim i*
shows $i = supr\ (underS\ i)$
⟨*proof*⟩

definition $pred\ i \equiv SOME\ j.\ j \in Field\ r \wedge aboveS\ j \neq \{\} \wedge succ\ j = i$

lemma *pred-Field-succ*:
assumes *isSucc i* **shows** $pred\ i \in Field\ r \wedge aboveS\ (pred\ i) \neq \{\} \wedge succ\ (pred\ i) = i$
⟨*proof*⟩

lemmas *pred-Field*[*simp*] = *pred-Field-succ*[*THEN conjunct1*]
lemmas *aboveS-pred*[*simp*] = *pred-Field-succ*[*THEN conjunct2*, *THEN conjunct1*]
lemmas *succ-pred*[*simp*] = *pred-Field-succ*[*THEN conjunct2*, *THEN conjunct2*]

lemma *isSucc-pred-in*:
assumes *isSucc i* **shows** $(pred\ i,\ i) \in r$
⟨*proof*⟩

lemma *isSucc-pred-diff*:
assumes *isSucc i* **shows** $pred\ i \neq i$
⟨*proof*⟩

lemma *succ-inj*[*simp*]:
assumes $aboveS\ i \neq \{\}$ **and** $aboveS\ j \neq \{\}$
shows $succ\ i = succ\ j \longleftrightarrow i = j$
⟨*proof*⟩

lemma *pred-succ*[*simp*]:
assumes $aboveS\ j \neq \{\}$ **shows** $pred\ (succ\ j) = j$
⟨*proof*⟩

lemma *less-succ*[*simp*]:
assumes $aboveS\ i \neq \{\}$
shows $(j,\ succ\ i) \in r \longleftrightarrow (j,i) \in r \vee j = succ\ i$

$\langle proof \rangle$

lemma *underS-succ*[*simp*]:
assumes *aboveS* $i \neq \{\}$
shows *underS* (*succ* i) = *under* i
$\langle proof \rangle$

lemma *succ-mono*:
assumes *aboveS* $j \neq \{\}$ **and** $(i,j) \in r$
shows (*succ* i, *succ* j) $\in r$
$\langle proof \rangle$

lemma *under-succ*[*simp*]:
assumes *aboveS* $i \neq \{\}$
shows *under* (*succ* i) = *insert* (*succ* i) (*under* i)
$\langle proof \rangle$

definition *mergeSL* :: $('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$
where
mergeSL S L f i $\equiv$ *if isSucc* i *then* S (*pred* i) (f (*pred* i)) *else* L f i

8.5.2 Well-order recursion with (zero), succesor, and limit

definition *worecSL* :: $('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$
where *worecSL* S L $\equiv$ *worec* (*mergeSL* S L)

definition *adm-woL* $L \equiv \forall f\ g\ i.$ *isLim* $i \wedge (\forall j \in$ *underS* $i.\ f\ j = g\ j) \longrightarrow L\ f\ i = L\ g\ i$

lemma *mergeSL*: *adm-woL* $L \Longrightarrow$ *adm-wo* (*mergeSL* S L)
$\langle proof \rangle$

lemma *worec-fixpoint1*: *adm-wo* $H \Longrightarrow$ *worec* H i = H (*worec* H) i
$\langle proof \rangle$

lemma *worecSL-isSucc*:
assumes *a*: *adm-woL* L **and** *i*: *isSucc* i
shows *worecSL* S L i = S (*pred* i) (*worecSL* S L (*pred* i))
$\langle proof \rangle$

lemma *worecSL-succ*:
assumes *a*: *adm-woL* L **and** *i*: *aboveS* $j \neq \{\}$
shows *worecSL* S L (*succ* j) = S j (*worecSL* S L j)
$\langle proof \rangle$

lemma *worecSL-isLim*:
assumes *a*: *adm-woL* L **and** *i*: *isLim* i
shows *worecSL* S L i = L (*worecSL* S L) i

$\langle proof \rangle$

definition *worecZSL* :: $'b \Rightarrow ('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$
where $worecZSL\ Z\ S\ L \equiv worecSL\ S\ (\lambda\ f\ a.\ if\ a = zero\ then\ Z\ else\ L\ f\ a)$

lemma *worecZSL-zero*:
assumes a: $adm\text{-}woL\ L$
shows $worecZSL\ Z\ S\ L\ zero = Z$
$\langle proof \rangle$

lemma *worecZSL-succ*:
assumes a: $adm\text{-}woL\ L$ **and** i: $aboveS\ j \neq \{\}$
shows $worecZSL\ Z\ S\ L\ (succ\ j) = S\ j\ (worecZSL\ Z\ S\ L\ j)$
$\langle proof \rangle$

lemma *worecZSL-isLim*:
assumes a: $adm\text{-}woL\ L$ **and** $isLim\ i$ **and** $i \neq zero$
shows $worecZSL\ Z\ S\ L\ i = L\ (worecZSL\ Z\ S\ L)\ i$
$\langle proof \rangle$

8.5.3 Well-order succ-lim induction

lemma *ord-cases*:
obtains j **where** $i = succ\ j$ **and** $aboveS\ j \neq \{\}$ $\mid isLim\ i$
$\langle proof \rangle$

lemma *well-order-inductSL*[*case-names Suc Lim*]:
assumes $\bigwedge i.\ [\![aboveS\ i \neq \{\};\ P\ i]\!] \Longrightarrow P\ (succ\ i)$ $\bigwedge i.\ [\![isLim\ i;\ \bigwedge j.\ j \in underS\ i \Longrightarrow P\ j]\!] \Longrightarrow P\ i$
shows $P\ i$
$\langle proof \rangle$

lemma *well-order-inductZSL*[*case-names Zero Suc Lim*]:
assumes $P\ zero$
and $\bigwedge i.\ [\![aboveS\ i \neq \{\};\ P\ i]\!] \Longrightarrow P\ (succ\ i)$ **and**
$\bigwedge i.\ [\![isLim\ i;\ i \neq zero;\ \bigwedge j.\ j \in underS\ i \Longrightarrow P\ j]\!] \Longrightarrow P\ i$
shows $P\ i$
$\langle proof \rangle$

definition $isSuccOrd \equiv \exists\ j \in Field\ r.\ \forall\ i \in Field\ r.\ (i,j) \in r$
definition $isLimOrd \equiv \neg\ isSuccOrd$

lemma *isLimOrd-succ*:
assumes $isLimOrd$ **and** $i \in Field\ r$
shows $succ\ i \in Field\ r$
$\langle proof \rangle$

lemma *isLimOrd-aboveS*:

assumes *l*: *isLimOrd* **and** *i*: $i \in$ *Field r*
shows *aboveS* $i \neq \{\}$
⟨*proof*⟩

lemma *succ-aboveS-isLimOrd*:
assumes $\forall$ $i \in$ *Field r*. *aboveS* $i \neq \{\} \wedge$ *succ* $i \in$ *Field r*
shows *isLimOrd*
⟨*proof*⟩

lemma *isLim-iff*:
assumes *l*: *isLim i* **and** *j*: $j \in$ *underS i*
shows $\exists$ *k*. $k \in$ *underS i* $\wedge$ $j \in$ *underS k*
⟨*proof*⟩

end

abbreviation *zero* $\equiv$ *wo-rel.zero*
abbreviation *succ* $\equiv$ *wo-rel.succ*
abbreviation *pred* $\equiv$ *wo-rel.pred*
abbreviation *isSucc* $\equiv$ *wo-rel.isSucc*
abbreviation *isLim* $\equiv$ *wo-rel.isLim*
abbreviation *isLimOrd* $\equiv$ *wo-rel.isLimOrd*
abbreviation *isSuccOrd* $\equiv$ *wo-rel.isSuccOrd*
abbreviation *adm-woL* $\equiv$ *wo-rel.adm-woL*
abbreviation *worecSL* $\equiv$ *wo-rel.worecSL*
abbreviation *worecZSL* $\equiv$ *wo-rel.worecZSL*

8.6 Projections of wellorders

definition *oproj r s f* $\equiv$ *Field s* $\subseteq$ *f* ‘ (*Field r*) $\wedge$ *compat r s f*

lemma *oproj-in*:
assumes *oproj r s f* **and** $(a,a') \in r$
shows $(f\ a,\ f\ a') \in s$
⟨*proof*⟩

lemma *oproj-Field*:
assumes *f*: *oproj r s f* **and** *a*: $a \in$ *Field r*
shows $f\ a \in$ *Field s*
⟨*proof*⟩

lemma *oproj-Field2*:
assumes *f*: *oproj r s f* **and** *a*: $b \in$ *Field s*
shows $\exists$ $a \in$ *Field r*. $f\ a = b$
⟨*proof*⟩

lemma *oproj-under*:
assumes *f*: *oproj r s f* **and** *a*: $a \in$ *under r a'*
shows $f\ a \in$ *under s* $(f\ a')$

$\langle proof \rangle$

theorem *embedI*:
assumes *r*: *Well-order r* **and** *s*: *Well-order s*
and *f*: $\bigwedge$ *a*. $a \in$ *Field r* $\Longrightarrow$ *f a* $\in$ *Field s* $\wedge$ *f* ‘ *underS r a* $\subseteq$ *underS s* (*f a*)
shows $\exists$ *g*. *embed r s g*
$\langle proof \rangle$

corollary *ordLeq-def2*:
$r \le o\ s \longleftrightarrow$ *Well-order r* $\wedge$ *Well-order s* $\wedge$
($\exists$ *f*. $\forall$ $a \in$ *Field r*. *f a* $\in$ *Field s* $\wedge$ *f* ‘ *underS r a* $\subseteq$ *underS s* (*f a*))
$\langle proof \rangle$

lemma *iso-oproj*:
assumes *r*: *Well-order r* **and** *s*: *Well-order s* **and** *f*: *iso r s f*
shows *oproj r s f*
$\langle proof \rangle$

theorem *oproj-embed*:
assumes *r*: *Well-order r* **and** *s*: *Well-order s* **and** *f*: *oproj r s f*
shows $\exists$ *g*. *embed s r g*
$\langle proof \rangle$

corollary *oproj-ordLeq*:
assumes *r*: *Well-order r* **and** *s*: *Well-order s* **and** *f*: *oproj r s f*
shows $s \le o\ r$
$\langle proof \rangle$

end

9 Ordinal Arithmetic

theory *Ordinal-Arithmetic*
imports *Wellorder-Constructions*
begin

definition *osum* :: $'a$ *rel* $\Rightarrow$ $'b$ *rel* $\Rightarrow$ ($'a$ + $'b$) *rel* (**infixr** +*o* *70*)
where
$r +o\ r'$ = *map-prod Inl Inl* ‘ *r* $\cup$ *map-prod Inr Inr* ‘ r' $\cup$
{(*Inl a*, *Inr a'*) | *a a'* . $a \in$ *Field r* $\wedge$ $a' \in$ *Field r'*}

lemma *Field-osum*: *Field*($r +o\ r'$) = *Inl* ‘ *Field r* $\cup$ *Inr* ‘ *Field r'*
$\langle proof \rangle$

lemma *osum-Refl*:⟦*Refl r*; *Refl r'*⟧ $\Longrightarrow$ *Refl* ($r +o\ r'$)

$\langle proof \rangle$

lemma *osum-trans*:
assumes *TRANS*: *trans r* **and** *TRANS'*: *trans r'*
shows *trans* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-Preorder*: $[\![$*Preorder r*; *Preorder r'*$]\!] \Longrightarrow$ *Preorder* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-antisym*: $[\![$*antisym r*; *antisym r'*$]\!] \Longrightarrow$ *antisym* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-Partial-order*: $[\![$*Partial-order r*; *Partial-order r'*$]\!] \Longrightarrow$ *Partial-order* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-Total*: $[\![$*Total r*; *Total r'*$]\!] \Longrightarrow$ *Total* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-Linear-order*: $[\![$*Linear-order r*; *Linear-order r'*$]\!] \Longrightarrow$ *Linear-order* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-wf*:
assumes *WF*: *wf r* **and** *WF'*: *wf r'*
shows *wf* $(r\ {+o}\ r')$
⟨*proof*⟩

lemma *osum-minus-Id*:
assumes *r*: *Total r* $\neg$ $(r \le Id)$ **and** *r'*: *Total r'* $\neg$ $(r' \le Id)$
shows $(r\ {+o}\ r') - Id \le (r - Id)\ {+o}\ (r' - Id)$
⟨*proof*⟩

lemma *osum-minus-Id1*:
$r \le Id \Longrightarrow (r\ {+o}\ r') - Id \le (Inl\ \text{‘}\ Field\ r \times Inr\ \text{‘}\ Field\ r') \cup (map\text{-}prod\ Inr\ Inr\ \text{‘}\ (r' - Id))$
⟨*proof*⟩

lemma *osum-minus-Id2*:
$r' \le Id \Longrightarrow (r\ {+o}\ r') - Id \le (map\text{-}prod\ Inl\ Inl\ \text{‘}\ (r - Id)) \cup (Inl\ \text{‘}\ Field\ r \times Inr\ \text{‘}\ Field\ r')$
⟨*proof*⟩

lemma *osum-wf-Id*:
assumes *TOT*: *Total r* **and** *TOT'*: *Total r'* **and** *WF*: $wf(r - Id)$ **and** *WF'*: $wf(r' - Id)$
shows *wf* $((r\ {+o}\ r') - Id)$
⟨*proof*⟩

lemma *osum-Well-order*:

assumes *WELL*: *Well-order r* **and** *WELL'*: *Well-order r'*
shows *Well-order* $(r +o\ r')$
⟨*proof*⟩

lemma *osum-embedL*:
assumes *WELL*: *Well-order r* **and** *WELL'*: *Well-order r'*
shows *embed r* $(r +o\ r')$ *Inl*
⟨*proof*⟩

corollary *osum-ordLeqL*:
assumes *WELL*: *Well-order r* **and** *WELL'*: *Well-order r'*
shows $r \leq o\ r +o\ r'$
⟨*proof*⟩

lemma *dir-image-alt*: *dir-image r f* = *map-prod f f* ‘ *r*
⟨*proof*⟩

lemma *map-prod-ordIso*: ⟦*Well-order r*; *inj-on f* (*Field r*)⟧ ⟹ *map-prod f f* ‘ *r* $=o\ r$
⟨*proof*⟩

definition *oprod* :: *'a rel* ⇒ *'b rel* ⇒ (*'a* × *'b*) *rel* (**infixr** $*o$ *80*)
where $r *o\ r' = \{((x1, y1), (x2, y2)).$
$(((y1, y2) \in r' - Id \wedge x1 \in Field\ r \wedge x2 \in Field\ r) \vee$
$((y1, y2) \in Restr\ Id\ (Field\ r') \wedge (x1, x2) \in r))\}$

lemma *Field-oprod*: *Field* $(r *o\ r')$ = *Field r* × *Field r'*
⟨*proof*⟩

lemma *oprod-Refl*:⟦*Refl r*; *Refl r'*⟧ ⟹ *Refl* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-trans*:
assumes *trans r trans r' antisym r antisym r'*
shows *trans* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-Preorder*: ⟦*Preorder r*; *Preorder r'*; *antisym r*; *antisym r'*⟧ ⟹ *Preorder* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-antisym*: ⟦*antisym r*; *antisym r'*⟧ ⟹ *antisym* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-Partial-order*: ⟦*Partial-order r*; *Partial-order r'*⟧ ⟹ *Partial-order* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-Total*: ⟦*Total r*; *Total r'*⟧ ⟹ *Total* $(r *o\ r')$

⟨*proof*⟩

lemma *oprod-Linear-order*: $[\![$*Linear-order* r; *Linear-order* $r'$$]\!] \Longrightarrow$ *Linear-order* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-wf*:
assumes *WF*: *wf* r **and** *WF′*: *wf* r'
shows *wf* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-minus-Id*:
assumes r: *Total* r $\neg$ $(r \leq Id)$ **and** r': *Total* r' $\neg$ $(r' \leq Id)$
shows $(r *o\ r') - Id \leq (r - Id) *o\ (r' - Id)$
⟨*proof*⟩

lemma *oprod-minus-Id1*:
$r \leq Id \Longrightarrow r *o\ r' - Id \leq \{((x,y1), (x,y2)).\ x \in Field\ r \wedge (y1, y2) \in (r' - Id)\}$
⟨*proof*⟩

lemma *wf-extend-oprod1*:
assumes *wf* r
shows *wf* $\{((x,y1), (x,y2))\ .\ x \in A \wedge (y1, y2) \in r\}$
⟨*proof*⟩

lemma *oprod-minus-Id2*:
$r' \leq Id \Longrightarrow r *o\ r' - Id \leq \{((x1,y), (x2,y)).\ (x1, x2) \in (r - Id) \wedge y \in Field\ r'\}$
⟨*proof*⟩

lemma *wf-extend-oprod2*:
assumes *wf* r
shows *wf* $\{((x1,y), (x2,y))\ .\ (x1, x2) \in r \wedge y \in A\}$
⟨*proof*⟩

lemma *oprod-wf-Id*:
assumes *TOT*: *Total* r **and** *TOT′*: *Total* r' **and** *WF*: $wf(r - Id)$ **and** *WF′*: $wf(r' - Id)$
shows *wf* $((r *o\ r') - Id)$
⟨*proof*⟩

lemma *oprod-Well-order*:
assumes *WELL*: *Well-order* r **and** *WELL′*: *Well-order* r'
shows *Well-order* $(r *o\ r')$
⟨*proof*⟩

lemma *oprod-embed*:
assumes *WELL*: *Well-order* r **and** *WELL′*: *Well-order* r' **and** $r' \neq \{\}$
shows *embed* r $(r *o\ r')$ $(\lambda x.\ (x,\ minim\ r'\ (Field\ r')))$ (**is** *embed - - ?f*)

$\langle proof \rangle$

corollary *oprod-ordLeq*: $[\![Well\text{-}order\ r;\ Well\text{-}order\ r';\ r' \neq \{\}]\!] \Longrightarrow r \leq o\ r *o\ r'$
$\langle proof \rangle$

definition *support* $z\ A\ f = \{x \in A.\ f\ x \neq z\}$

lemma *support-Un*[*simp*]: $support\ z\ (A \cup B)\ f = support\ z\ A\ f \cup support\ z\ B\ f$
$\langle proof \rangle$

lemma *support-upd*[*simp*]: $support\ z\ A\ (f(x := z)) = support\ z\ A\ f - \{x\}$
$\langle proof \rangle$

lemma *support-upd-subset*[*simp*]: $support\ z\ A\ (f(x := y)) \subseteq support\ z\ A\ f \cup \{x\}$
$\langle proof \rangle$

lemma *fun-unequal-in-support*:
assumes $f \neq g\ f \in Func\ A\ B\ g \in Func\ A\ C$
shows $(support\ z\ A\ f \cup support\ z\ A\ g) \cap \{a.\ f\ a \neq g\ a\} \neq \{\}$
$\langle proof \rangle$

definition *fin-support* **where**
$fin\text{-}support\ z\ A = \{f.\ finite\ (support\ z\ A\ f)\}$

lemma *finite-support*: $f \in fin\text{-}support\ z\ A \Longrightarrow finite\ (support\ z\ A\ f)$
$\langle proof \rangle$

lemma *fin-support-Field-osum*:
$f \in fin\text{-}support\ z\ (Inl\ `\ A \cup Inr\ `\ B) \longleftrightarrow$
$(f\ o\ Inl) \in fin\text{-}support\ z\ A \wedge (f\ o\ Inr) \in fin\text{-}support\ z\ B$ (**is** $?L \longleftrightarrow ?R1 \wedge ?R2$)
$\langle proof \rangle$

lemma *Func-upd*: $[\![f \in Func\ A\ B;\ x \in A;\ y \in B]\!] \Longrightarrow f(x := y) \in Func\ A\ B$
$\langle proof \rangle$

context *wo-rel*
begin

definition *isMaxim* :: $'a\ set \Rightarrow 'a \Rightarrow bool$
where $isMaxim\ A\ b \equiv b \in A \wedge (\forall\, a \in A.\ (a,b) \in r)$

definition *maxim* :: $'a\ set \Rightarrow 'a$
where $maxim\ A \equiv THE\ b.\ isMaxim\ A\ b$

lemma *isMaxim-unique*[*intro*]: $[\![isMaxim\ A\ x;\ isMaxim\ A\ y]\!] \Longrightarrow x = y$
$\langle proof \rangle$

lemma *maxim-isMaxim*: $[\![finite\ A;\ A \neq \{\};\ A \subseteq Field\ r]\!] \Longrightarrow isMaxim\ A\ (maxim\ A)$

⟨*proof*⟩

lemma *maxim-in*: $[\![$*finite* A; $A \neq \{\}$; $A \subseteq$ *Field* $r]\!] \Longrightarrow$ *maxim* $A \in A$
⟨*proof*⟩

lemma *maxim-greatest*: $[\![$*finite* A; $x \in A$; $A \subseteq$ *Field* $r]\!] \Longrightarrow (x,$ *maxim* $A) \in r$
⟨*proof*⟩

lemma *isMaxim-zero*: *isMaxim* A *zero* $\Longrightarrow A = \{$*zero*$\}$
⟨*proof*⟩

lemma *maxim-insert*:
assumes *finite* A $A \neq \{\}$ $A \subseteq$ *Field* r $x \in$ *Field* r
shows *maxim* (*insert* x A) = *max2* x (*maxim* A)
⟨*proof*⟩

lemma *maxim-Un*:
assumes *finite* A $A \neq \{\}$ $A \subseteq$ *Field* r *finite* B $B \neq \{\}$ $B \subseteq$ *Field* r
shows *maxim* $(A \cup B)$ = *max2* (*maxim* A) (*maxim* B)
⟨*proof*⟩

lemma *maxim-insert-zero*:
assumes *finite* A $A \neq \{\}$ $A \subseteq$ *Field* r
shows *maxim* (*insert zero* A) = *maxim* A
⟨*proof*⟩

lemma *maxim-equality*: *isMaxim* A $x \Longrightarrow$ *maxim* $A = x$
⟨*proof*⟩

lemma *maxim-singleton*:
$x \in$ *Field* $r \Longrightarrow$ *maxim* $\{x\} = x$
⟨*proof*⟩

lemma *maxim-Int*: $[\![$*finite* A; $A \neq \{\}$; $A \subseteq$ *Field* r; *maxim* $A \in B]\!] \Longrightarrow$ *maxim* $(A \cap B)$ = *maxim* A
⟨*proof*⟩

lemma *maxim-mono*: $[\![X \subseteq Y$; *finite* Y; $X \neq \{\}$; $Y \subseteq$ *Field* $r]\!] \Longrightarrow$ (*maxim* X, *maxim* $Y) \in r$
⟨*proof*⟩

definition *max-fun-diff* f $g \equiv$ *maxim* $(\{a \in$ *Field* $r.$ f $a \neq g$ $a\})$

lemma *max-fun-diff-commute*: *max-fun-diff* f g = *max-fun-diff* g f
⟨*proof*⟩

lemma *zero-under*: $x \in$ *Field* $r \Longrightarrow$ *zero* $\in$ *under* x
⟨*proof*⟩

end

definition *FinFunc r s* = *Func* (*Field s*) (*Field r*) ∩ *fin-support* (*zero r*) (*Field s*)

lemma *FinFuncD*: ⟦*f* ∈ *FinFunc r s*; *x* ∈ *Field s*⟧ ⟹ *f x* ∈ *Field r*
⟨*proof*⟩

locale *wo-rel2* =
fixes *r s*
assumes *rWELL*: *Well-order r*
and *sWELL*: *Well-order s*
begin

interpretation *r*: *wo-rel r* ⟨*proof*⟩
interpretation *s*: *wo-rel s* ⟨*proof*⟩

abbreviation *SUPP* ≡ *support r.zero* (*Field s*)
abbreviation *FINFUNC* ≡ *FinFunc r s*
lemmas *FINFUNCD* = *FinFuncD*[*of* - *r s*]

lemma *fun-diff-alt*: {*a* ∈ *Field s*. *f a* ≠ *g a*} = (*SUPP f* ∪ *SUPP g*) ∩ {*a*. *f a* ≠ *g a*}
⟨*proof*⟩

lemma *max-fun-diff-alt*:
s.max-fun-diff f g = *s.maxim* ((*SUPP f* ∪ *SUPP g*) ∩ {*a*. *f a* ≠ *g a*})
⟨*proof*⟩

lemma *isMaxim-max-fun-diff*: ⟦*f* ≠ *g*; *f* ∈ *FINFUNC*; *g* ∈ *FINFUNC*⟧ ⟹
s.isMaxim {*a* ∈ *Field s*. *f a* ≠ *g a*} (*s.max-fun-diff f g*)
⟨*proof*⟩

lemma *max-fun-diff-in*: ⟦*f* ≠ *g*; *f* ∈ *FINFUNC*; *g* ∈ *FINFUNC*⟧ ⟹
s.max-fun-diff f g ∈ {*a* ∈ *Field s*. *f a* ≠ *g a*}
⟨*proof*⟩

lemma *max-fun-diff-max*: ⟦*f* ≠ *g*; *f* ∈ *FINFUNC*; *g* ∈ *FINFUNC*; *x* ∈ {*a* ∈ *Field s*. *f a* ≠ *g a*}⟧ ⟹
(*x*, *s.max-fun-diff f g*) ∈ *s*
⟨*proof*⟩

lemma *max-fun-diff*:
⟦*f* ≠ *g*; *f* ∈ *FINFUNC*; *g* ∈ *FINFUNC*⟧ ⟹
(∃ *a b*. *a* ≠ *b* ∧ *a* ∈ *Field r* ∧ *b* ∈ *Field r* ∧
f (*s.max-fun-diff f g*) = *a* ∧ *g* (*s.max-fun-diff f g*) = *b*)
⟨*proof*⟩

lemma *max-fun-diff-le-eq*:
⟦(*s.max-fun-diff f g*, *x*) ∈ *s*; *f* ≠ *g*; *f* ∈ *FINFUNC*; *g* ∈ *FINFUNC*; *x* ≠ *s.max-fun-diff*

$f\ g\rrbracket \Longrightarrow$
  $f\ x = g\ x$
  ⟨*proof*⟩

lemma *max-fun-diff-max2*:
  assumes *ineq*: *s.max-fun-diff f g = s.max-fun-diff g h* $\longrightarrow$
    f (*s.max-fun-diff f g*) $\neq$ *h* (*s.max-fun-diff g h*) **and**
    fg: $f \neq g$ **and** *gh*: $g \neq h$ **and** *fh*: $f \neq h$ **and**
    f: $f \in$ *FINFUNC* **and** *g*: $g \in$ *FINFUNC* **and** *h*: $h \in$ *FINFUNC*
  shows *s.max-fun-diff f h = s.max2* (*s.max-fun-diff f g*) (*s.max-fun-diff g h*)
    (**is** *?fh = s.max2 ?fg ?gh*)
⟨*proof*⟩

definition *oexp* **where**
  oexp = {(*f*, *g*) . *f* ∈ *FINFUNC* ∧ *g* ∈ *FINFUNC* ∧
    ((*let m = s.max-fun-diff f g in* (*f m*, *g m*) ∈ *r*) ∨ *f* = *g*)}

lemma *Field-oexp*: *Field oexp = FINFUNC*
  ⟨*proof*⟩

lemma *oexp-Refl*: *Refl oexp*
  ⟨*proof*⟩

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

lemma *oexp-Preorder*: *Preorder oexp*
  ⟨*proof*⟩

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

lemma *oexp-Partial-order*: *Partial-order oexp*
  ⟨*proof*⟩

lemma *oexp-Total*: *Total oexp*
  ⟨*proof*⟩

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

definition *const* = (λ*x*. *if x* ∈ *Field s then r.zero else undefined*)

lemma *const-in*[*simp*]: *x* ∈ *Field s* $\Longrightarrow$ *const x = r.zero*
  ⟨*proof*⟩

lemma *const-notin*[*simp*]: *x* ∉ *Field s* $\Longrightarrow$ *const x = undefined*
  ⟨*proof*⟩

lemma *const-Int-Field*[*simp*]: $Field\ s \cap - \{x.\ const\ x = r.zero\} = \{\}$
⟨*proof*⟩

lemma *const-FINFUNC*[*simp*]: $Field\ r \neq \{\} \Longrightarrow const \in FINFUNC$
⟨*proof*⟩

lemma *const-least*:
assumes $Field\ r \neq \{\}\ f \in FINFUNC$
shows $(const,\ f) \in oexp$
⟨*proof*⟩

lemma *support-not-const*:
assumes $F \subseteq FINFUNC$ **and** $const \notin F$
shows $\forall f \in F.\ finite\ (SUPP\ f) \wedge SUPP\ f \neq \{\} \wedge SUPP\ f \subseteq Field\ s$
⟨*proof*⟩

lemma *maxim-isMaxim-support*:
assumes $F \subseteq FINFUNC$ **and** $const \notin F$
shows $\forall f \in F.\ s.isMaxim\ (SUPP\ f)\ (s.maxim\ (SUPP\ f))$
⟨*proof*⟩

lemma *oexp-empty2*: $Field\ s = \{\} \Longrightarrow oexp = \{(\lambda x.\ undefined,\ \lambda x.\ undefined)\}$
⟨*proof*⟩

lemma *oexp-empty*: $[\![Field\ r = \{\};\ Field\ s \neq \{\}]\!] \Longrightarrow oexp = \{\}$
⟨*proof*⟩

lemma *fun-upd-FINFUNC*: $[\![f \in FINFUNC;\ x \in Field\ s;\ y \in Field\ r]\!] \Longrightarrow f(x := y) \in FINFUNC$
⟨*proof*⟩

lemma *fun-upd-same-oexp*:
assumes $(f,\ g) \in oexp\ f\ x = g\ x\ x \in Field\ s\ y \in Field\ r$
shows $(f(x := y),\ g(x := y)) \in oexp$
⟨*proof*⟩

lemma *fun-upd-smaller-oexp*:
assumes $f \in FINFUNC\ x \in Field\ s\ y \in Field\ r\ \ (y,\ f\ x) \in r$
shows $(f(x := y),\ f) \in oexp$
⟨*proof*⟩

lemma *oexp-wf-Id*: $wf\ (oexp - Id)$
⟨*proof*⟩

lemma *oexp-Well-order*: *Well-order oexp*
⟨*proof*⟩

interpretation *o*: *wo-rel oexp* ⟨*proof*⟩

lemma *zero-oexp*: $Field\ r \neq \{\} \Longrightarrow o.zero = const$
⟨*proof*⟩

end

notation *wo-rel2.oexp* (**infixl** $\hat{}o$ *90*)
lemmas *oexp-def* = *wo-rel2.oexp-def*[*unfolded wo-rel2-def*, *OF conjI*]
lemmas *oexp-Well-order* = *wo-rel2.oexp-Well-order*[*unfolded wo-rel2-def*, *OF conjI*]
lemmas *Field-oexp* = *wo-rel2.Field-oexp*[*unfolded wo-rel2-def*, *OF conjI*]

definition *ozero* = {}

lemma *ozero-Well-order*[*simp*]: *Well-order ozero*
⟨*proof*⟩

lemma *ozero-ordIso*[*simp*]: *ozero =o ozero*
⟨*proof*⟩

lemma *Field-ozero*[*simp*]: *Field ozero* = {}
⟨*proof*⟩

lemma *iso-ozero-empty*[*simp*]: *r =o ozero* = (*r* = {})
⟨*proof*⟩

lemma *ozero-ordLeq*:
assumes *Well-order r* **shows** $ozero \leq o\ r$
⟨*proof*⟩

definition *oone* = {((),())}

lemma *oone-Well-order*[*simp*]: *Well-order oone*
⟨*proof*⟩

lemma *Field-oone*[*simp*]: *Field oone* = {()}
⟨*proof*⟩

lemma *oone-ordIso*: *oone =o* {(*x*,*x*)}
⟨*proof*⟩

lemma *osum-ordLeqR*: $Well\text{-}order\ r \Longrightarrow Well\text{-}order\ s \Longrightarrow s \leq o\ r +o\ s$
⟨*proof*⟩

lemma *osum-congL*:
assumes *r =o s* **and** *t*: *Well-order t*
shows *r +o t =o s +o t* (**is** *?L =o ?R*)
⟨*proof*⟩

lemma *osum-congR*:
assumes *r =o s* **and** *t*: *Well-order t*

shows t $+o$ r $=o$ t $+o$ s (**is** $?L$ $=o$ $?R$)
⟨*proof*⟩

lemma *osum-cong*:
assumes t $=o$ u **and** r $=o$ s
shows t $+o$ r $=o$ u $+o$ s
⟨*proof*⟩

lemma *Well-order-empty*[*simp*]: *Well-order* $\{\}$
⟨*proof*⟩

lemma *well-order-on-singleton*[*simp*]: *well-order-on* $\{x\}$ $\{(x, x)\}$
⟨*proof*⟩

lemma *oexp-empty*[*simp*]:
assumes *Well-order* r
shows r $\hat{}o$ $\{\}$ $=$ $\{(\lambda x.$ *undefined*, $\lambda x.$ *undefined*$)\}$
⟨*proof*⟩

lemma *oexp-empty2*[*simp*]:
assumes *Well-order* r $r \neq \{\}$
shows $\{\}$ $\hat{}o$ r $=$ $\{\}$
⟨*proof*⟩

lemma *oprod-zero*[*simp*]: $\{\}$ $*o$ r $=$ $\{\}$ r $*o$ $\{\}$ $=$ $\{\}$
⟨*proof*⟩

lemma *oprod-congL*:
assumes r $=o$ s **and** t: *Well-order* t
shows r $*o$ t $=o$ s $*o$ t (**is** $?L$ $=o$ $?R$)
⟨*proof*⟩

lemma *oprod-congR*:
assumes r $=o$ s **and** t: *Well-order* t
shows t $*o$ r $=o$ t $*o$ s (**is** $?L$ $=o$ $?R$)
⟨*proof*⟩

lemma *oprod-cong*:
assumes t $=o$ u **and** r $=o$ s
shows t $*o$ r $=o$ u $*o$ s
⟨*proof*⟩

lemma *Field-singleton*[*simp*]: *Field* $\{(z,z)\}$ $=$ $\{z\}$
⟨*proof*⟩

lemma *zero-singleton*[*simp*]: *zero* $\{(z,z)\}$ $=$ z
⟨*proof*⟩

lemma *FinFunc-singleton*: *FinFunc* $\{(z,z)\}$ s $=$ $\{\lambda x.$ *if* $x \in$ *Field* s *then* z *else*

undefined}
⟨*proof*⟩

lemma *oone-ordIso-oexp*:
assumes $r =o$ *oone* **and** *s*: *Well-order s*
shows r ^o $s =o$ *oone* (**is** *?L =o ?R*)
⟨*proof*⟩

context
fixes *r s t*
assumes *r*: *Well-order r*
assumes *s*: *Well-order s*
assumes *t*: *Well-order t*
begin

lemma *osum-ozeroL*: *ozero* $+o$ $r =o$ r
⟨*proof*⟩

lemma *osum-ozeroR*: r $+o$ *ozero* $=o$ r
⟨*proof*⟩

lemma *osum-assoc*: $(r +o s) +o t =o r +o s +o t$ (**is** *?L =o ?R*)
⟨*proof*⟩

lemma *osum-monoR*:
assumes $s <o t$
shows $r +o s <o r +o t$ (**is** *?L <o ?R*)
⟨*proof*⟩

lemma *osum-monoL*:
assumes $r \leq o\ s$
shows $r +o t \leq o\ s +o t$
⟨*proof*⟩

lemma *oprod-ozeroL*: *ozero* $*o$ $r =o$ *ozero*
⟨*proof*⟩

lemma *oprod-ozeroR*: r $*o$ *ozero* $=o$ *ozero*
⟨*proof*⟩

lemma *oprod-ooneR*: r $*o$ *oone* $=o$ r (**is** *?L =o ?R*)
⟨*proof*⟩

lemma *oprod-ooneL*: *oone* $*o$ $r =o$ r (**is** *?L =o ?R*)
⟨*proof*⟩

lemma *oprod-monoR*:
assumes *ozero* $<o$ r $s <o t$

shows $r *o\ s <o\ r *o\ t$ (**is** $?L <o\ ?R$)
⟨*proof*⟩

lemma *oprod-monoL*:
assumes $r \leq o\ s$
shows $r *o\ t \leq o\ s *o\ t$
⟨*proof*⟩

lemma *oprod-assoc*: $(r *o\ s) *o\ t =o\ r *o\ s *o\ t$ (**is** $?L =o\ ?R$)
⟨*proof*⟩

lemma *oprod-osum*: $r *o\ (s +o\ t) =o\ r *o\ s +o\ r *o\ t$ (**is** $?L =o\ ?R$)
⟨*proof*⟩

lemma *ozero-oexp*: $\neg\ (s =o\ ozero) \Longrightarrow ozero\ \hat{}o\ s =o\ ozero$
⟨*proof*⟩

lemma *oone-oexp*: $oone\ \hat{}o\ s =o\ oone$ (**is** $?L =o\ ?R$)
⟨*proof*⟩

lemma *oexp-monoR*:
assumes $oone <o\ r\ s <o\ t$
shows $r\ \hat{}o\ s <o\ r\ \hat{}o\ t$ (**is** $?L <o\ ?R$)
⟨*proof*⟩

lemma *oexp-monoL*:
assumes $r \leq o\ s$
shows $r\ \hat{}o\ t \leq o\ s\ \hat{}o\ t$
⟨*proof*⟩

lemma *ordLeq-oexp2*:
assumes $oone <o\ r$
shows $s \leq o\ r\ \hat{}o\ s$
⟨*proof*⟩

lemma *FinFunc-osum*:
$fg \in FinFunc\ r\ (s +o\ t) = (fg\ o\ Inl \in FinFunc\ r\ s \wedge fg\ o\ Inr \in FinFunc\ r\ t)$
(**is** $?L = (?R1 \wedge ?R2)$)
⟨*proof*⟩

lemma *max-fun-diff-eq-Inl*:
assumes *wo-rel.max-fun-diff* $(s +o\ t)$ (*case-sum f1 g1*) (*case-sum f2 g2*) $= Inl\ x$
case-sum f1 g1 $\neq$ *case-sum f2 g2*
case-sum f1 g1 $\in FinFunc\ r\ (s +o\ t)$ *case-sum f2 g2* $\in FinFunc\ r\ (s +o\ t)$
shows *wo-rel.max-fun-diff s f1 f2* $= x$ (**is** $?P$) $g1 = g2$ (**is** $?Q$)
⟨*proof*⟩

lemma *max-fun-diff-eq-Inr*:
assumes *wo-rel.max-fun-diff* $(s +o\ t)$ (*case-sum f1 g1*) (*case-sum f2 g2*) $= Inr$

x
  case-sum f1 g1 $\neq$ *case-sum f2 g2*
  case-sum f1 g1 $\in$ *FinFunc r* $(s +o\ t)$ *case-sum f2 g2* $\in$ *FinFunc r* $(s +o\ t)$
shows *wo-rel.max-fun-diff t g1 g2* $= x$ (**is** *?P*) *g1* $\neq$ *g2* (**is** *?Q*)
⟨*proof*⟩

lemma *oexp-osum*: r ^o $(s +o\ t) =o\ (r$ ^$o\ s) *o\ (r$ ^$o\ t)$ (**is** *?R =o ?L*)
⟨*proof*⟩

definition *rev-curr f b* = (*if b* $\in$ *Field t then* $\lambda a.\ f\ (a,\ b)$ *else undefined*)

lemma *rev-curr-FinFunc*:
assumes *Field*: *Field r* $\neq$ {}
shows *rev-curr* ‘ (*FinFunc r* $(s *o\ t)$) = *FinFunc* $(r$ ^$o\ s)\ t$
⟨*proof*⟩

lemma *rev-curr-app-FinFunc*[*elim*!]:
⟦$f \in$ *FinFunc r* $(s *o\ t)$; $z \in$ *Field t*⟧ $\Longrightarrow$ *rev-curr f z* $\in$ *FinFunc r s*
⟨*proof*⟩

lemma *max-fun-diff-oprod*:
assumes *Field*: *Field r* $\neq$ {} **and** $f \neq g$ $f \in$ *FinFunc r* $(s *o\ t)$ $g \in$ *FinFunc r* $(s *o\ t)$
defines $m \equiv$ *wo-rel.max-fun-diff t* (*rev-curr f*) (*rev-curr g*)
shows *wo-rel.max-fun-diff* $(s *o\ t)$ *f g* =
  (*wo-rel.max-fun-diff s* (*rev-curr f m*) (*rev-curr g m*), *m*)
⟨*proof*⟩

lemma *oexp-oexp*: $(r$ ^$o\ s)$ ^$o\ t =o\ r$ ^$o\ (s *o\ t)$ (**is** *?R =o ?L*)
⟨*proof*⟩

end

end

10 Cardinal-Order Relations

theory *Cardinal-Order-Relation*
imports *Wellorder-Constructions*
begin

declare
card-order-on-well-order-on[*simp*]
card-of-card-order-on[*simp*]
card-of-well-order-on[*simp*]
Field-card-of[*simp*]
card-of-Card-order[*simp*]
card-of-Well-order[*simp*]
card-of-least[*simp*]

card-of-unique[*simp*]
card-of-mono1[*simp*]
card-of-mono2[*simp*]
card-of-cong[*simp*]
card-of-Field-ordIso[*simp*]
card-of-underS[*simp*]
ordLess-Field[*simp*]
card-of-empty[*simp*]
card-of-empty1[*simp*]
card-of-image[*simp*]
card-of-singl-ordLeq[*simp*]
Card-order-singl-ordLeq[*simp*]
card-of-Pow[*simp*]
Card-order-Pow[*simp*]
card-of-Plus1[*simp*]
Card-order-Plus1[*simp*]
card-of-Plus2[*simp*]
Card-order-Plus2[*simp*]
card-of-Plus-mono1[*simp*]
card-of-Plus-mono2[*simp*]
card-of-Plus-mono[*simp*]
card-of-Plus-cong2[*simp*]
card-of-Plus-cong[*simp*]
card-of-Un-Plus-ordLeq[*simp*]
card-of-Times1[*simp*]
card-of-Times2[*simp*]
card-of-Times3[*simp*]
card-of-Times-mono1[*simp*]
card-of-Times-mono2[*simp*]
card-of-ordIso-finite[*simp*]
card-of-Times-same-infinite[*simp*]
card-of-Times-infinite-simps[*simp*]
card-of-Plus-infinite1[*simp*]
card-of-Plus-infinite2[*simp*]
card-of-Plus-ordLess-infinite[*simp*]
card-of-Plus-ordLess-infinite-Field[*simp*]
infinite-cartesian-product[*simp*]
cardSuc-Card-order[*simp*]
cardSuc-greater[*simp*]
cardSuc-ordLeq[*simp*]
cardSuc-ordLeq-ordLess[*simp*]
cardSuc-mono-ordLeq[*simp*]
cardSuc-invar-ordIso[*simp*]
card-of-cardSuc-finite[*simp*]
cardSuc-finite[*simp*]
card-of-Plus-ordLeq-infinite-Field[*simp*]
curr-in[*intro*, *simp*]

10.1 Cardinal of a set

lemma *card-of-inj-rel*: **assumes** *INJ*: $\bigwedge x\ y\ y'.\ [\![(x,y) \in R;\ (x,y') \in R]\!] \Longrightarrow y = y'$
shows $|\{y.\ \exists x.\ (x,y) \in R\}|$ <=o $|\{x.\ \exists y.\ (x,y) \in R\}|$
⟨*proof*⟩

lemma *card-of-unique2*: $[\![$*card-order-on* $B\ r$; *bij-betw* $f\ A\ B]\!] \Longrightarrow r$ =o $|A|$
⟨*proof*⟩

lemma *internalize-card-of-ordLess2*:
$(\ |A|$ <o $|C|\ ) = (\exists B < C.\ |A|$ =o $|B| \wedge |B|$ <o $|C|\ )$
⟨*proof*⟩

lemma *Card-order-omax*:
assumes *finite* R **and** $R \neq \{\}$ **and** $\forall r{\in}R.$ *Card-order* r
shows *Card-order* (*omax* R)
⟨*proof*⟩

lemma *Card-order-omax2*:
assumes *finite* I **and** $I \neq \{\}$
shows *Card-order* (*omax* $\{|A\ i|\ |\ i.\ i \in I\}$)
⟨*proof*⟩

10.2 Cardinals versus set operations on arbitrary sets

lemma *card-of-set-type*[*simp*]: $|UNIV::'a\ set|$ <o $|UNIV::'a\ set\ set|$
⟨*proof*⟩

lemma *card-of-Un1*[*simp*]: $|A| \leq o\ |A \cup B|$
⟨*proof*⟩

lemma *card-of-diff*[*simp*]: $|A - B| \leq o\ |A|$
⟨*proof*⟩

lemma *subset-ordLeq-strict*:
assumes $A \leq B$ **and** $|A|$ <o $|B|$
shows $A < B$
⟨*proof*⟩

corollary *subset-ordLeq-diff*:
assumes $A \leq B$ **and** $|A|$ <o $|B|$
shows $B - A \neq \{\}$
⟨*proof*⟩

lemma *card-of-empty4*:
$|\{\}::'b\ set|$ <o $|A::'a\ set| = (A \neq \{\})$
⟨*proof*⟩

lemma *card-of-empty5*:
$|A|$ <o $|B| \Longrightarrow B \neq \{\}$

$\langle proof \rangle$

lemma *Well-order-card-of-empty*:
$Well\text{-}order\ r \Longrightarrow |\{\}| \leq o\ r$
$\langle proof \rangle$

lemma *card-of-UNIV*[*simp*]:
$|A :: 'a\ set| \leq o\ |UNIV :: 'a\ set|$
$\langle proof \rangle$

lemma *card-of-UNIV2*[*simp*]:
$Card\text{-}order\ r \Longrightarrow (r :: 'a\ rel) \leq o\ |UNIV :: 'a\ set|$
$\langle proof \rangle$

lemma *card-of-Pow-mono*[*simp*]:
assumes $|A| \leq o\ |B|$
shows $|Pow\ A| \leq o\ |Pow\ B|$
$\langle proof \rangle$

lemma *ordIso-Pow-mono*[*simp*]:
assumes $r \leq o\ r'$
shows $|Pow(Field\ r)| \leq o\ |Pow(Field\ r')|$
$\langle proof \rangle$

lemma *card-of-Pow-cong*[*simp*]:
assumes $|A| =o\ |B|$
shows $|Pow\ A| =o\ |Pow\ B|$
$\langle proof \rangle$

lemma *ordIso-Pow-cong*[*simp*]:
assumes $r =o\ r'$
shows $|Pow(Field\ r)| =o\ |Pow(Field\ r')|$
$\langle proof \rangle$

corollary *Card-order-Plus-empty1*:
$Card\text{-}order\ r \Longrightarrow r =o\ |(Field\ r) <+> \{\}|$
$\langle proof \rangle$

corollary *Card-order-Plus-empty2*:
$Card\text{-}order\ r \Longrightarrow r =o\ |\{\} <+> (Field\ r)|$
$\langle proof \rangle$

lemma *card-of-Un2*[*simp*]:
shows $|A| \leq o\ |B \cup A|$
$\langle proof \rangle$

lemma *Un-Plus-bij-betw*:
assumes $A\ Int\ B = \{\}$
shows $\exists f.\ bij\text{-}betw\ f\ (A \cup B)\ (A <+> B)$

⟨*proof*⟩

lemma *card-of-Un-Plus-ordIso*:
assumes A *Int* $B = \{\}$
shows $|A \cup B| =o |A <+> B|$
⟨*proof*⟩

lemma *card-of-Un-Plus-ordIso1*:
$|A \cup B| =o |A <+> (B - A)|$
⟨*proof*⟩

lemma *card-of-Un-Plus-ordIso2*:
$|A \cup B| =o |(A - B) <+> B|$
⟨*proof*⟩

lemma *card-of-Times-singl1*: $|A| =o |A \times \{b\}|$
⟨*proof*⟩

corollary *Card-order-Times-singl1*:
Card-order $r \Longrightarrow r =o |(Field\ r) \times \{b\}|$
⟨*proof*⟩

lemma *card-of-Times-singl2*: $|A| =o |\{b\} \times A|$
⟨*proof*⟩

corollary *Card-order-Times-singl2*:
Card-order $r \Longrightarrow r =o |\{a\} \times (Field\ r)|$
⟨*proof*⟩

lemma *card-of-Times-assoc*: $|(A \times B) \times C| =o |A \times B \times C|$
⟨*proof*⟩

lemma *card-of-Times-cong1*[*simp*]:
assumes $|A| =o |B|$
shows $|A \times C| =o |B \times C|$
⟨*proof*⟩

lemma *card-of-Times-cong2*[*simp*]:
assumes $|A| =o |B|$
shows $|C \times A| =o |C \times B|$
⟨*proof*⟩

lemma *card-of-Times-mono*[*simp*]:
assumes $|A| \leq o |B|$ **and** $|C| \leq o |D|$
shows $|A \times C| \leq o |B \times D|$
⟨*proof*⟩

corollary *ordLeq-Times-mono*:
assumes $r \leq o r'$ **and** $p \leq o p'$

shows $|(Field\ r) \times (Field\ p)| \leq o\ |(Field\ r') \times (Field\ p')|$
⟨*proof*⟩

corollary *ordIso-Times-cong1* [*simp*]:
assumes $r =o\ r'$
shows $|(Field\ r) \times C| =o\ |(Field\ r') \times C|$
⟨*proof*⟩

corollary *ordIso-Times-cong2*:
assumes $r =o\ r'$
shows $|A \times (Field\ r)| =o\ |A \times (Field\ r')|$
⟨*proof*⟩

lemma *card-of-Times-cong*[*simp*]:
assumes $|A| =o\ |B|$ **and** $|C| =o\ |D|$
shows $|A \times C| =o\ |B \times D|$
⟨*proof*⟩

corollary *ordIso-Times-cong*:
assumes $r =o\ r'$ **and** $p =o\ p'$
shows $|(Field\ r) \times (Field\ p)| =o\ |(Field\ r') \times (Field\ p')|$
⟨*proof*⟩

lemma *card-of-Sigma-mono2*:
assumes *inj-on f* $(I::'i\ set)$ **and** $f\ `\ I \leq (J::'j\ set)$
shows $|SIGMA\ i : I.\ (A::'j \Rightarrow 'a\ set)\ (f\ i)| \leq o\ |SIGMA\ j : J.\ A\ j|$
⟨*proof*⟩

lemma *card-of-Sigma-mono*:
assumes *INJ*: *inj-on f I* **and** *IM*: $f\ `\ I \leq J$ **and**
LEQ: $\forall j \in J.\ |A\ j| \leq o\ |B\ j|$
shows $|SIGMA\ i : I.\ A\ (f\ i)| \leq o\ |SIGMA\ j : J.\ B\ j|$
⟨*proof*⟩

lemma *ordLeq-Sigma-mono1*:
assumes $\forall i \in I.\ p\ i \leq o\ r\ i$
shows $|SIGMA\ i : I.\ Field(p\ i)| \leq o\ |SIGMA\ i : I.\ Field(r\ i)|$
⟨*proof*⟩

lemma *ordLeq-Sigma-mono*:
assumes *inj-on f I* **and** $f\ `\ I \leq J$ **and**
$\forall j \in J.\ p\ j \leq o\ r\ j$
shows $|SIGMA\ i : I.\ Field(p(f\ i))| \leq o\ |SIGMA\ j : J.\ Field(r\ j)|$
⟨*proof*⟩

lemma *ordIso-Sigma-cong1*:
assumes $\forall i \in I.\ p\ i =o\ r\ i$
shows $|SIGMA\ i : I.\ Field(p\ i)| =o\ |SIGMA\ i : I.\ Field(r\ i)|$
⟨*proof*⟩

lemma *ordLeq-Sigma-cong*:
assumes *bij-betw f I J* **and**
$\forall j \in J.\ p\ j =o\ r\ j$
shows $|SIGMA\ i : I.\ Field(p(f\ i))| =o\ |SIGMA\ j : J.\ Field(r\ j)|$
⟨*proof*⟩

lemma *card-of-UNION-Sigma2*:
assumes $\bigwedge i\ j.\ [\![\{i,j\} <= I;\ i \neq j]\!] \Longrightarrow A\ i\ Int\ A\ j = \{\}$
shows $|\bigcup i{\in}I.\ A\ i| =o\ |Sigma\ I\ A|$
⟨*proof*⟩

corollary *Plus-into-Times*:
assumes $A2$: $a1 \neq a2 \wedge \{a1,a2\} \leq A$ **and** $B2$: $b1 \neq b2 \wedge \{b1,b2\} \leq B$
shows $\exists f.\ inj\text{-}on\ f\ (A <+> B) \wedge f\ `\ (A <+> B) \leq A \times B$
⟨*proof*⟩

corollary *Plus-into-Times-types*:
assumes $A2$: $(a1::'a) \neq a2$ **and** $B2$: $(b1::'b) \neq b2$
shows $\exists (f::'a + 'b \Rightarrow 'a * 'b).\ inj\ f$
⟨*proof*⟩

corollary *Times-same-infinite-bij-betw*:
assumes $\neg finite\ A$
shows $\exists f.\ bij\text{-}betw\ f\ (A \times A)\ A$
⟨*proof*⟩

corollary *Times-same-infinite-bij-betw-types*:
assumes INF: $\neg finite(UNIV::'a\ set)$
shows $\exists (f::('a * 'a) => 'a).\ bij\ f$
⟨*proof*⟩

corollary *Times-infinite-bij-betw*:
assumes INF: $\neg finite\ A$ **and** NE: $B \neq \{\}$ **and** INJ: $inj\text{-}on\ g\ B \wedge g\ `\ B \leq A$
shows $(\exists f.\ bij\text{-}betw\ f\ (A \times B)\ A) \wedge (\exists h.\ bij\text{-}betw\ h\ (B \times A)\ A)$
⟨*proof*⟩

corollary *Times-infinite-bij-betw-types*:
assumes $\neg finite(UNIV::'a\ set)$ **and** $inj(g::'b \Rightarrow 'a)$
shows $(\exists (f::('b * 'a) => 'a).\ bij\ f) \wedge (\exists (h::('a * 'b) => 'a).\ bij\ h)$
⟨*proof*⟩

lemma *card-of-Times-ordLeq-infinite*:
$[\![\neg finite\ C;\ |A| \leq o\ |C|;\ |B| \leq o\ |C|]\!] \Longrightarrow |A \times B| \leq o\ |C|$
⟨*proof*⟩

corollary *Plus-infinite-bij-betw*:
assumes INF: $\neg finite\ A$ **and** INJ: $inj\text{-}on\ g\ B \wedge g\ `\ B \leq A$
shows $(\exists f.\ bij\text{-}betw\ f\ (A <+> B)\ A) \wedge (\exists h.\ bij\text{-}betw\ h\ (B <+> A)\ A)$

⟨proof⟩

corollary *Plus-infinite-bij-betw-types*:
assumes $\neg$*finite*(*UNIV*::′*a set*) **and** *inj*(g::′$b \Rightarrow$ ′a)
shows $(\exists (f::('b + 'a) => 'a).\ bij\ f) \wedge (\exists (h::('a + 'b) => 'a).\ bij\ h)$
⟨proof⟩

lemma *card-of-Un-infinite*:
assumes *INF*: $\neg$*finite* A **and** *LEQ*: $|B| \leq o\ |A|$
shows $|A \cup B| =o\ |A| \wedge |B \cup A| =o\ |A|$
⟨proof⟩

lemma *card-of-Un-infinite-simps*[*simp*]:
$[\![\neg finite\ A;\ |B| \leq o\ |A|\ ]\!] \Longrightarrow |A \cup B| =o\ |A|$
$[\![\neg finite\ A;\ |B| \leq o\ |A|\ ]\!] \Longrightarrow |B \cup A| =o\ |A|$
⟨proof⟩

lemma *card-of-Un-diff-infinite*:
assumes *INF*: $\neg$*finite* A **and** *LESS*: $|B| <o\ |A|$
shows $|A - B| =o\ |A|$
⟨proof⟩

corollary *Card-order-Un-infinite*:
assumes *INF*: $\neg$*finite*(*Field* r) **and** *CARD*: *Card-order* r **and**
LEQ: $p \leq o\ r$
shows $|\ (Field\ r) \cup (Field\ p)\ | =o\ r \wedge |\ (Field\ p) \cup (Field\ r)\ | =o\ r$
⟨proof⟩

corollary *subset-ordLeq-diff-infinite*:
assumes *INF*: $\neg$*finite* B **and** *SUB*: $A \leq B$ **and** *LESS*: $|A| <o\ |B|$
shows $\neg$*finite* $(B - A)$
⟨proof⟩

lemma *card-of-Times-ordLess-infinite*[*simp*]:
assumes *INF*: $\neg$*finite* C **and**
LESS1: $|A| <o\ |C|$ **and** *LESS2*: $|B| <o\ |C|$
shows $|A \times B| <o\ |C|$
⟨proof⟩

lemma *card-of-Times-ordLess-infinite-Field*[*simp*]:
assumes *INF*: $\neg$*finite* (*Field* r) **and** r: *Card-order* r **and**
LESS1: $|A| <o\ r$ **and** *LESS2*: $|B| <o\ r$
shows $|A \times B| <o\ r$
⟨proof⟩

lemma *ordLeq-finite-Field*:
assumes $r \leq o\ s$ **and** *finite* (*Field* s)
shows *finite* (*Field* r)
⟨proof⟩

lemma *ordIso-finite-Field*:
  assumes $r =o\ s$
  shows *finite* (*Field r*) $\longleftrightarrow$ *finite* (*Field s*)
  ⟨*proof*⟩

10.3 Cardinals versus set operations involving infinite sets

lemma *finite-iff-cardOf-nat*:
  finite $A = (\ |A| <o\ |UNIV :: nat\ set|\ )$
  ⟨*proof*⟩

lemma *finite-ordLess-infinite2* [*simp*]:
  assumes *finite A* **and** ¬*finite B*
  shows $|A| <o\ |B|$
  ⟨*proof*⟩

lemma *infinite-card-of-insert*:
  assumes ¬*finite A*
  shows $|insert\ a\ A| =o\ |A|$
⟨*proof*⟩

lemma *card-of-Un-singl-ordLess-infinite1*:
  assumes ¬*finite B* **and** $|A| <o\ |B|$
  shows $|\{a\}\ Un\ A| <o\ |B|$
  ⟨*proof*⟩

lemma *card-of-Un-singl-ordLess-infinite*:
  assumes ¬*finite B*
  shows $|A| <o\ |B| \longleftrightarrow |\{a\}\ Un\ A| <o\ |B|$
  ⟨*proof*⟩

10.4 Cardinals versus lists

The next is an auxiliary operator, which shall be used for inductive proofs of facts concerning the cardinality of *List* :

definition *nlists* :: $'a\ set \Rightarrow nat \Rightarrow 'a\ list\ set$
  where $nlists\ A\ n \equiv \{l.\ set\ l \leq A \wedge length\ l = n\}$

lemma *lists-UNION-nlists*: $lists\ A = (\bigcup n.\ nlists\ A\ n)$
  ⟨*proof*⟩

lemma *card-of-lists*: $|A| \leq o\ |lists\ A|$
⟨*proof*⟩

lemma *nlists-0*: $nlists\ A\ 0 = \{[]\}$
  ⟨*proof*⟩

lemma *nlists-not-empty*:

assumes $A \neq \{\}$
shows *nlists* $A\ n \neq \{\}$
⟨*proof*⟩

lemma *card-of-nlists-Succ*: $|nlists\ A\ (Suc\ n)| =o\ |A \times (nlists\ A\ n)|$
⟨*proof*⟩

lemma *card-of-nlists-infinite*:
assumes $\neg finite\ A$
shows $|nlists\ A\ n| \leq o\ |A|$
⟨*proof*⟩

lemma *Card-order-lists*: $Card\text{-}order\ r \Longrightarrow r \leq o\ |lists(Field\ r)\ |$
⟨*proof*⟩

lemma *Union-set-lists*: $\bigcup(set\ `\ (lists\ A)) = A$
⟨*proof*⟩

lemma *inj-on-map-lists*:
assumes *inj-on f A*
shows *inj-on* $(map\ f)\ (lists\ A)$
⟨*proof*⟩

lemma *map-lists-mono*:
assumes $f\ `\ A \leq B$
shows $(map\ f)\ `\ (lists\ A) \leq lists\ B$
⟨*proof*⟩

lemma *map-lists-surjective*:
assumes $f\ `\ A = B$
shows $(map\ f)\ `\ (lists\ A) = lists\ B$
⟨*proof*⟩

lemma *bij-betw-map-lists*:
assumes *bij-betw f A B*
shows *bij-betw* $(map\ f)\ (lists\ A)\ (lists\ B)$
⟨*proof*⟩

lemma *card-of-lists-mono*[*simp*]:
assumes $|A| \leq o\ |B|$
shows $|lists\ A| \leq o\ |lists\ B|$
⟨*proof*⟩

lemma *ordIso-lists-mono*:
assumes $r \leq o\ r'$
shows $|lists(Field\ r)| \leq o\ |lists(Field\ r')|$
⟨*proof*⟩

lemma *card-of-lists-cong*[*simp*]:
  assumes $|A| =o |B|$
  shows $|lists\ A| =o |lists\ B|$
  ⟨*proof*⟩

lemma *card-of-lists-infinite*[*simp*]:
  assumes $\neg finite\ A$
  shows $|lists\ A| =o |A|$
⟨*proof*⟩

lemma *Card-order-lists-infinite*:
  assumes *Card-order r* **and** $\neg finite(Field\ r)$
  shows $|lists(Field\ r)| =o r$
  ⟨*proof*⟩

lemma *ordIso-lists-cong*:
  assumes $r =o r'$
  shows $|lists(Field\ r)| =o |lists(Field\ r')|$
  ⟨*proof*⟩

corollary *lists-infinite-bij-betw*:
  assumes $\neg finite\ A$
  shows $\exists f.\ bij\text{-}betw\ f\ (lists\ A)\ A$
  ⟨*proof*⟩

corollary *lists-infinite-bij-betw-types*:
  assumes $\neg finite(UNIV :: 'a\ set)$
  shows $\exists (f::'a\ list \Rightarrow 'a).\ bij\ f$
  ⟨*proof*⟩

10.5 Cardinals versus the finite powerset operator

lemma *card-of-Fpow*[*simp*]: $|A| \leq o\ |Fpow\ A|$
⟨*proof*⟩

lemma *Card-order-Fpow*: *Card-order r* $\Longrightarrow r \leq o\ |Fpow(Field\ r)\ |$
  ⟨*proof*⟩

lemma *image-Fpow-surjective*:
  assumes $f\ `\ A = B$
  shows $(image\ f)\ `\ (Fpow\ A) = Fpow\ B$
⟨*proof*⟩

lemma *bij-betw-image-Fpow*:
  assumes *bij-betw f A B*
  shows *bij-betw* (*image f*) (*Fpow A*) (*Fpow B*)
  ⟨*proof*⟩

lemma *card-of-Fpow-mono*[*simp*]:

assumes $|A| \le o\ |B|$
shows $|Fpow\ A| \le o\ |Fpow\ B|$
⟨*proof*⟩

lemma *ordIso-Fpow-mono*:
assumes $r \le o\ r'$
shows $|Fpow(Field\ r)| \le o\ |Fpow(Field\ r')|$
⟨*proof*⟩

lemma *card-of-Fpow-cong*[*simp*]:
assumes $|A| =o\ |B|$
shows $|Fpow\ A| =o\ |Fpow\ B|$
⟨*proof*⟩

lemma *ordIso-Fpow-cong*:
assumes $r =o\ r'$
shows $|Fpow(Field\ r)| =o\ |Fpow(Field\ r')|$
⟨*proof*⟩

lemma *card-of-Fpow-lists*: $|Fpow\ A| \le o\ |lists\ A|$
⟨*proof*⟩

lemma *card-of-Fpow-infinite*[*simp*]:
assumes $\neg finite\ A$
shows $|Fpow\ A| =o\ |A|$
⟨*proof*⟩

corollary *Fpow-infinite-bij-betw*:
assumes $\neg finite\ A$
shows $\exists f.\ bij\text{-}betw\ f\ (Fpow\ A)\ A$
⟨*proof*⟩

10.6 The cardinal ω and the finite cardinals

10.6.1 First as well-orders

lemma *Field-natLess*: *Field natLess* = (*UNIV*::*nat set*)
⟨*proof*⟩

lemma *natLeq-well-order-on*: *well-order-on UNIV natLeq*
⟨*proof*⟩

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

lemma *natLeq-ofilter-less*: *ofilter natLeq* $\{0\ ..<\ n\}$
⟨*proof*⟩

lemma *natLeq-ofilter-leq*: *ofilter natLeq* $\{0\ ..\ n\}$
⟨*proof*⟩

lemma *natLeq-UNIV-ofilter*: *wo-rel.ofilter natLeq UNIV*
⟨*proof*⟩

lemma *closed-nat-set-iff*:
assumes $\forall (m::nat)\ n.\ n \in A \wedge m \leq n \longrightarrow m \in A$
shows $A = UNIV \vee (\exists n.\ A = \{0\ ..< n\})$
⟨*proof*⟩

lemma *natLeq-ofilter-iff*:
$ofilter\ natLeq\ A = (A = UNIV \vee (\exists n.\ A = \{0\ ..< n\}))$
⟨*proof*⟩

lemma *natLeq-under-leq*: $under\ natLeq\ n = \{0\ ..\ n\}$
⟨*proof*⟩

lemma *natLeq-on-ofilter-less-eq*:
$n \leq m \Longrightarrow wo\text{-}rel.ofilter\ (natLeq\text{-}on\ m)\ \{0\ ..< n\}$
⟨*proof*⟩

lemma *natLeq-on-ofilter-iff*:
$wo\text{-}rel.ofilter\ (natLeq\text{-}on\ m)\ A = (\exists n \leq m.\ A = \{0\ ..< n\})$
⟨*proof*⟩

corollary *natLeq-on-ofilter*:
$ofilter(natLeq\text{-}on\ n)\ \{0\ ..< n\}$
⟨*proof*⟩

lemma *natLeq-on-ofilter-less*:
assumes $n < m$ **shows** $ofilter\ (natLeq\text{-}on\ m)\ \{0\ ..\ n\}$
⟨*proof*⟩

lemma *natLeq-on-ordLess-natLeq*: $natLeq\text{-}on\ n <o\ natLeq$
⟨*proof*⟩

lemma *natLeq-on-injective*:
$natLeq\text{-}on\ m = natLeq\text{-}on\ n \Longrightarrow m = n$
⟨*proof*⟩

lemma *natLeq-on-injective-ordIso*:
$(natLeq\text{-}on\ m =o\ natLeq\text{-}on\ n) = (m = n)$
⟨*proof*⟩

10.6.2 Then as cardinals

lemma *ordIso-natLeq-infinite1*:
$|A| =o\ natLeq \Longrightarrow \neg finite\ A$
⟨*proof*⟩

lemma *ordIso-natLeq-infinite2*:
$natLeq =o |A| \Longrightarrow \neg finite\ A$
⟨*proof*⟩

lemma *ordIso-natLeq-on-imp-finite*:
$|A| =o\ natLeq\text{-}on\ n \Longrightarrow finite\ A$
⟨*proof*⟩

lemma *natLeq-on-Card-order*: $Card\text{-}order\ (natLeq\text{-}on\ n)$
⟨*proof*⟩

corollary *card-of-Field-natLeq-on*:
$|Field\ (natLeq\text{-}on\ n)| =o\ natLeq\text{-}on\ n$
⟨*proof*⟩

corollary *card-of-less*:
$|\{0\ ..< n\}| =o\ natLeq\text{-}on\ n$
⟨*proof*⟩

lemma *natLeq-on-ordLeq-less-eq*:
$((natLeq\text{-}on\ m) \le o\ (natLeq\text{-}on\ n)) = (m \le n)$
⟨*proof*⟩

lemma *natLeq-on-ordLeq-less*:
$((natLeq\text{-}on\ m) <o\ (natLeq\text{-}on\ n)) = (m < n)$
⟨*proof*⟩

lemma *ordLeq-natLeq-on-imp-finite*:
assumes $|A| \le o\ natLeq\text{-}on\ n$
shows *finite A*
⟨*proof*⟩

10.6.3 "Backward compatibility" with the numeric cardinal operator for finite sets

lemma *finite-card-of-iff-card2*:
assumes *FIN*: *finite A* **and** *FIN'*: *finite B*
shows $(\ |A| \le o\ |B|\ ) = (card\ A \le card\ B)$
⟨*proof*⟩

lemma *finite-imp-card-of-natLeq-on*:
assumes *finite A*
shows $|A| =o\ natLeq\text{-}on\ (card\ A)$
⟨*proof*⟩

lemma *finite-iff-card-of-natLeq-on*:
$finite\ A = (\exists n.\ |A| =o\ natLeq\text{-}on\ n)$
⟨*proof*⟩

lemma *finite-card-of-iff-card*:
assumes *FIN*: *finite A* **and** *FIN'*: *finite B*
shows ($|A| =o |B|$) = (*card A* = *card B*)
⟨*proof*⟩

lemma *finite-card-of-iff-card3*:
assumes *FIN*: *finite A* **and** *FIN'*: *finite B*
shows ($|A| <o |B|$) = (*card A* < *card B*)
⟨*proof*⟩

lemma *card-Field-natLeq-on*:
card(*Field*(*natLeq-on n*)) = *n*
⟨*proof*⟩

10.7 The successor of a cardinal

lemma *embed-implies-ordIso-Restr*:
assumes *WELL*: *Well-order r* **and** *WELL'*: *Well-order r'* **and** *EMB*: *embed r' r f*
shows *r'* =*o Restr r* (*f* ‘ (*Field r'*))
⟨*proof*⟩

lemma *cardSuc-mono-ordLess*[*simp*]:
assumes *CARD*: *Card-order r* **and** *CARD'*: *Card-order r'*
shows (*cardSuc r* <*o cardSuc r'*) = (*r* <*o r'*)
⟨*proof*⟩

lemma *cardSuc-natLeq-on-Suc*:
cardSuc(*natLeq-on n*) =*o natLeq-on*(*Suc n*)
⟨*proof*⟩

lemma *card-of-Plus-ordLeq-infinite*[*simp*]:
assumes ¬*finite C* **and** $|A| \leq o |C|$ **and** $|B| \leq o |C|$
shows $|A <+> B| \leq o |C|$
⟨*proof*⟩

lemma *card-of-Un-ordLeq-infinite*[*simp*]:
assumes ¬*finite C* **and** $|A| \leq o |C|$ **and** $|B| \leq o |C|$
shows $|A\ Un\ B| \leq o |C|$
⟨*proof*⟩

10.8 Others

lemma *under-mono*[*simp*]:
assumes *Well-order r* **and** (*i*,*j*) ∈ *r*
shows *under r i* ⊆ *under r j*
⟨*proof*⟩

lemma *underS-under*:
assumes *i* ∈ *Field r*

shows *underS r i* = *under r i* − {*i*}
⟨*proof*⟩

lemma *relChain-under*:
assumes *Well-order r*
shows *relChain r* (λ *i*. *under r i*)
⟨*proof*⟩

lemma *card-of-infinite-diff-finite*:
assumes ¬*finite A* **and** *finite B*
shows |*A* − *B*| =*o* |*A*|
⟨*proof*⟩

lemma *infinite-card-of-diff-singl*:
assumes ¬*finite A*
shows |*A* − {*a*}| =*o* |*A*|
⟨*proof*⟩

lemma *card-of-vimage*:
assumes *B* ⊆ *range f*
shows |*B*| ≤*o* |*f* −' *B*|
⟨*proof*⟩

lemma *surj-card-of-vimage*:
assumes *surj f*
shows |*B*| ≤*o* |*f* −' *B*|
⟨*proof*⟩

definition *Bpow* **where**
Bpow r A ≡ {*X* . *X* ⊆ *A* ∧ |*X*| ≤*o r*}

lemma *Bpow-empty*[*simp*]:
assumes *Card-order r*
shows *Bpow r* {} = {{}}
⟨*proof*⟩

lemma *singl-in-Bpow*:
assumes *rc*: *Card-order r*
and *r*: *Field r* ≠ {} **and** *a*: *a* ∈ *A*
shows {*a*} ∈ *Bpow r A*
⟨*proof*⟩

lemma *ordLeq-card-Bpow*:
assumes *rc*: *Card-order r* **and** *r*: *Field r* ≠ {}
shows |*A*| ≤*o* |*Bpow r A*|
⟨*proof*⟩

lemma *infinite-Bpow*:

assumes *rc*: *Card-order r* **and** *r*: *Field r* $\neq$ {}
and *A*: $\neg$*finite A*
shows $\neg$*finite* (*Bpow r A*)
⟨*proof*⟩

definition *Func-option* **where**
Func-option A B $\equiv$
{*f*. ($\forall$ *a*. *f a* $\neq$ *None* $\longleftrightarrow$ *a* $\in$ *A*) $\wedge$ ($\forall$ *a* $\in$ *A*. *case f a of Some b* $\Rightarrow$ *b* $\in$ *B* |*None* $\Rightarrow$ *True*)}

lemma *card-of-Func-option-Func*:
|*Func-option A B*| =*o* |*Func A B*|
⟨*proof*⟩

definition *Pfunc* **where**
Pfunc A B $\equiv$
{*f*. ($\forall$ *a*. *f a* $\neq$ *None* $\longrightarrow$ *a* $\in$ *A*) $\wedge$
($\forall$ *a*. *case f a of None* $\Rightarrow$ *True* | *Some b* $\Rightarrow$ *b* $\in$ *B*)}

lemma *Func-Pfunc*:
Func-option A B $\subseteq$ *Pfunc A B*
⟨*proof*⟩

lemma *Pfunc-Func-option*:
Pfunc A B = ($\bigcup$ *A'* $\in$ *Pow A*. *Func-option A' B*)
⟨*proof*⟩

lemma *card-of-Func-mono*:
fixes *A1 A2* :: *'a set* **and** *B* :: *'b set*
assumes *A12*: *A1* $\subseteq$ *A2* **and** *B*: *B* $\neq$ {}
shows |*Func A1 B*| $\leq$*o* |*Func A2 B*|
⟨*proof*⟩

lemma *card-of-Func-option-mono*:
fixes *A1 A2* :: *'a set* **and** *B* :: *'b set*
assumes *A12*: *A1* $\subseteq$ *A2* **and** *B*: *B* $\neq$ {}
shows |*Func-option A1 B*| $\leq$*o* |*Func-option A2 B*|
⟨*proof*⟩

lemma *card-of-Pfunc-Pow-Func-option*:
assumes *B* $\neq$ {}
shows |*Pfunc A B*| $\leq$*o* |*Pow A* $\times$ *Func-option A B*|
⟨*proof*⟩

lemma *Bpow-ordLeq-Func-Field*:
assumes *rc*: *Card-order r* **and** *r*: *Field r* $\neq$ {} **and** *A*: $\neg$*finite A*
shows |*Bpow r A*| $\leq$*o* |*Func* (*Field r*) *A*|
⟨*proof*⟩

lemma *empty-in-Func*[*simp*]:
$B \neq \{\} \Longrightarrow (\lambda x.\ undefined) \in Func\ \{\}\ B$
⟨*proof*⟩

lemma *Func-mono*[*simp*]:
assumes $B1 \subseteq B2$
shows $Func\ A\ B1 \subseteq Func\ A\ B2$
⟨*proof*⟩

lemma *Pfunc-mono*[*simp*]:
assumes $A1 \subseteq A2$ **and** $B1 \subseteq B2$
shows $Pfunc\ A\ B1 \subseteq Pfunc\ A\ B2$
⟨*proof*⟩

lemma *card-of-Func-UNIV-UNIV*:
$|Func\ (UNIV::'a\ set)\ (UNIV::'b\ set)| =o\ |UNIV::('a \Rightarrow 'b)\ set|$
⟨*proof*⟩

lemma *ordLeq-Func*:
assumes $\{b1,b2\} \subseteq B\ b1 \neq b2$
shows $|A| \leq o\ |Func\ A\ B|$
⟨*proof*⟩

lemma *infinite-Func*:
assumes A: $\neg finite\ A$ **and** B: $\{b1,b2\} \subseteq B\ b1 \neq b2$
shows $\neg finite\ (Func\ A\ B)$
⟨*proof*⟩

10.9 Infinite cardinals are limit ordinals

lemma *card-order-infinite-isLimOrd*:
assumes c: $Card\text{-}order\ r$ **and** i: $\neg finite\ (Field\ r)$
shows $isLimOrd\ r$
⟨*proof*⟩

lemma *insert-Chain*:
assumes $Refl\ r\ C \in Chains\ r$ **and** $i \in Field\ r$ **and** $\bigwedge j.\ j \in C \Longrightarrow (j,i) \in r \vee (i,j) \in r$
shows $insert\ i\ C \in Chains\ r$
⟨*proof*⟩

lemma *Collect-insert*: $\{R\ j\ |j.\ j \in insert\ j1\ J\} = insert\ (R\ j1)\ \{R\ j\ |j.\ j \in J\}$
⟨*proof*⟩

lemma *Field-init-seg-of*[*simp*]:
$Field\ init\text{-}seg\text{-}of = UNIV$
⟨*proof*⟩

lemma *refl-init-seg-of*[*intro*, *simp*]: *refl init-seg-of*
⟨*proof*⟩

lemma *regularCard-all-ex*:
assumes *r*: *Card-order r*   *regularCard r*
and *As*: $\bigwedge$ *i j b*. $b \in B \Longrightarrow (i,j) \in r \Longrightarrow P\ i\ b \Longrightarrow P\ j\ b$
and *Bsub*: $\forall\ b \in B.\ \exists\ i \in$ *Field r*. $P\ i\ b$
and *cardB*: $|B| <o\ r$
shows $\exists\ i \in$ *Field r*. $\forall\ b \in B.\ P\ i\ b$
⟨*proof*⟩

lemma *relChain-stabilize*:
assumes *rc*: *relChain r As* **and** *AsB*: $(\bigcup i \in$ *Field r*. *As i*$) \subseteq B$ **and** *Br*: $|B| <o$ r
and *ir*: ¬*finite* (*Field r*) **and** *cr*: *Card-order r*
shows $\exists\ i \in$ *Field r*. *As* (*succ r i*) = *As i*
⟨*proof*⟩

10.10 Regular vs. stable cardinals

lemma *stable-cardSuc*:
assumes *CARD*: *Card-order r* **and** *INF*: ¬*finite* (*Field r*)
shows *stable*(*cardSuc r*)
⟨*proof*⟩

lemma *stable-ordIso*:
assumes $r =o\ r'$
shows *stable r* = *stable r'*
⟨*proof*⟩

lemma *stable-nat*: *stable* |*UNIV*::*nat set*|
⟨*proof*⟩

Below, the type of "A" is not important – we just had to choose an appropriate type to make "A" possible. What is important is that arbitrarily large infinite sets of stable cardinality exist.

lemma *infinite-stable-exists*:
assumes *CARD*: $\forall r \in R$. *Card-order* (*r*::'*a rel*)
shows $\exists$(*A* :: (*nat* + '*a set*)*set*).
¬*finite A* ∧ *stable* |*A*| ∧ ($\forall r \in R.\ r <o\ |A|$)
⟨*proof*⟩

corollary *infinite-regularCard-exists*:
assumes *CARD*: $\forall r \in R$. *Card-order* (*r*::'*a rel*)
shows $\exists$(*A* :: (*nat* + '*a set*)*set*).
¬*finite A* ∧ *regularCard* |*A*| ∧ ($\forall r \in R.\ r <o\ |A|$)
⟨*proof*⟩

end

11 Cardinal Arithmetic

theory *Cardinal-Arithmetic*
imports *Cardinal-Order-Relation*
begin

11.1 Binary sum

lemma *csum-Cnotzero2*:
$Cnotzero\ r2 \Longrightarrow Cnotzero\ (r1\ +c\ r2)$
⟨*proof*⟩

lemma *single-cone*:
$|\{x\}|\ =o\ cone$
⟨*proof*⟩

lemma *cone-Cnotzero*: $Cnotzero\ cone$
⟨*proof*⟩

lemma *cone-ordLeq-ctwo*: $cone\ \leq o\ ctwo$
⟨*proof*⟩

lemma *csum-czero1*: $Card\text{-}order\ r \Longrightarrow r\ +c\ czero\ =o\ r$
⟨*proof*⟩

lemma *csum-czero2*: $Card\text{-}order\ r \Longrightarrow czero\ +c\ r\ =o\ r$
⟨*proof*⟩

11.2 Product

lemma *Times-cprod*: $|A \times B|\ =o\ |A|\ *c\ |B|$
⟨*proof*⟩

lemma *card-of-Times-singleton*:
fixes $A :: 'a\ set$
shows $|A \times \{x\}|\ =o\ |A|$
⟨*proof*⟩

lemma *cprod-assoc*: $(r\ *c\ s)\ *c\ t\ =o\ r\ *c\ s\ *c\ t$
⟨*proof*⟩

lemma *cprod-czero*: $r\ *c\ czero\ =o\ czero$
⟨*proof*⟩

lemma *cprod-cone*: $Card\text{-}order\ r \Longrightarrow r\ *c\ cone\ =o\ r$
⟨*proof*⟩

lemma *ordLeq-cprod1*: $[\![Card\text{-}order\ p1;\ Cnotzero\ p2]\!] \Longrightarrow p1\ \leq o\ p1\ *c\ p2$
⟨*proof*⟩

11.3 Exponentiation

lemma *cexp-czero*: $r\ \hat{}c\ czero =o\ cone$
  ⟨*proof*⟩

lemma *Pow-cexp-ctwo*:
  $|Pow\ A| =o\ ctwo\ \hat{}c\ |A|$
  ⟨*proof*⟩

lemma *Cnotzero-cexp*:
  assumes *Cnotzero q*
  shows *Cnotzero* $(q\ \hat{}c\ r)$
⟨*proof*⟩

lemma *Cinfinite-ctwo-cexp*:
  $Cinfinite\ r \Longrightarrow Cinfinite\ (ctwo\ \hat{}c\ r)$
  ⟨*proof*⟩

lemma *cone-ordLeq-iff-Field*:
  assumes $cone \le o\ r$
  shows $Field\ r \neq \{\}$
  ⟨*proof*⟩

lemma *cone-ordLeq-cexp*: $cone \le o\ r1 \Longrightarrow cone \le o\ r1\ \hat{}c\ r2$
  ⟨*proof*⟩

lemma *Card-order-czero*: *Card-order czero*
  ⟨*proof*⟩

lemma *cexp-mono2''*:
  assumes *2*: $p2 \le o\ r2$
    and *n1*: *Cnotzero q*
    and *n2*: *Card-order p2*
  shows $q\ \hat{}c\ p2 \le o\ q\ \hat{}c\ r2$
⟨*proof*⟩

lemma *csum-cexp*: $[\![Cinfinite\ r1;\ Cinfinite\ r2;\ Card\text{-}order\ q;\ ctwo \le o\ q]\!] \Longrightarrow$
  $q\ \hat{}c\ r1 +c\ q\ \hat{}c\ r2 \le o\ q\ \hat{}c\ (r1 +c\ r2)$
  ⟨*proof*⟩

lemma *csum-cexp'*: $[\![Cinfinite\ r;\ Card\text{-}order\ q;\ ctwo \le o\ q]\!] \Longrightarrow q +c\ r \le o\ q\ \hat{}c\ r$
  ⟨*proof*⟩

lemma *card-of-Sigma-ordLeq-Cinfinite*:
  $[\![Cinfinite\ r;\ |I| \le o\ r;\ \forall\, i \in I.\ |A\ i| \le o\ r]\!] \Longrightarrow |SIGMA\ i : I.\ A\ i| \le o\ r$
  ⟨*proof*⟩

lemma *Cinfinite-ordLess-cexp*:
  assumes *r*: *Cinfinite r*
  shows $r <o\ r\ \hat{}c\ r$

⟨*proof*⟩

lemma *infinite-ordLeq-cexp*:
assumes *Cinfinite r*
shows $r \leq o\ r\ \hat{}c\ r$
⟨*proof*⟩

lemma *czero-cexp*: *Cnotzero r* $\Longrightarrow$ *czero* $\hat{}c\ r =o$ *czero*
⟨*proof*⟩

lemma *Func-singleton*:
fixes $x :: 'b$ **and** $A :: 'a\ set$
shows $|Func\ A\ \{x\}| =o\ |\{x\}|$
⟨*proof*⟩

lemma *cone-cexp*: *cone* $\hat{}c\ r =o$ *cone*
⟨*proof*⟩

lemma *card-of-Func-squared*:
fixes $A :: 'a\ set$
shows $|Func\ (UNIV :: bool\ set)\ A| =o\ |A \times A|$
⟨*proof*⟩

lemma *cexp-ctwo*: $r\ \hat{}c\ ctwo =o\ r *c\ r$
⟨*proof*⟩

lemma *card-of-Func-Plus*:
fixes $A :: 'a\ set$ **and** $B :: 'b\ set$ **and** $C :: 'c\ set$
shows $|Func\ (A <+> B)\ C| =o\ |Func\ A\ C \times Func\ B\ C|$
⟨*proof*⟩

lemma *cexp-csum*: $r\ \hat{}c\ (s +c\ t) =o\ r\ \hat{}c\ s *c\ r\ \hat{}c\ t$
⟨*proof*⟩

11.4 Powerset

definition *cpow* **where** *cpow r* = $|Pow\ (Field\ r)|$

lemma *card-order-cpow*: *card-order r* $\Longrightarrow$ *card-order* (*cpow r*)
⟨*proof*⟩

lemma *cpow-greater-eq*: *Card-order r* $\Longrightarrow r \leq o$ *cpow r*
⟨*proof*⟩

lemma *Cinfinite-cpow*: *Cinfinite r* $\Longrightarrow$ *Cinfinite* (*cpow r*)
⟨*proof*⟩

lemma *Card-order-cpow*: *Card-order* (*cpow r*)
⟨*proof*⟩

lemma *cardSuc-ordLeq-cpow*: *Card-order r* $\Longrightarrow$ *cardSuc r* $\leq o$ *cpow r*
⟨*proof*⟩

lemma *cpow-cexp-ctwo*: *cpow r* $=o$ *ctwo* $\widehat{\ }c$ *r*
⟨*proof*⟩

11.5 Inverse image

lemma *vimage-ordLeq*:
assumes $|A| \leq o\ k$ **and** $\forall\ a \in A.\ |vimage\ f\ \{a\}| \leq o\ k$ **and** *Cinfinite k*
shows $|vimage\ f\ A| \leq o\ k$
⟨*proof*⟩

11.6 Maximum

definition *cmax* **where**
cmax r s =
(*if cinfinite r* $\vee$ *cinfinite s then czero* $+c$ *r* $+c$ *s*
else natLeq-on (*max* (*card* (*Field r*)) (*card* (*Field s*))) $+c$ *czero*)

lemma *cmax-com*: *cmax r s* $=o$ *cmax s r*
⟨*proof*⟩

lemma *cmax1*:
assumes *Card-order r Card-order s s* $\leq o$ *r*
shows *cmax r s* $=o$ *r*
⟨*proof*⟩

lemma *cmax2*:
assumes *Card-order r Card-order s r* $\leq o$ *s*
shows *cmax r s* $=o$ *s*
⟨*proof*⟩

context
fixes *r s*
assumes *r*: *Cinfinite r*
and *s*: *Cinfinite s*
begin

lemma *cmax-csum*: *cmax r s* $=o$ *r* $+c$ *s*
⟨*proof*⟩

lemma *cmax-cprod*: *cmax r s* $=o$ *r* $*c$ *s*
⟨*proof*⟩

end

lemma *Card-order-cmax*:
assumes *r*: *Card-order r* **and** *s*: *Card-order s*

shows *Card-order* (*cmax r s*)
⟨*proof*⟩

lemma *ordLeq-cmax*:
assumes *r*: *Card-order r* **and** *s*: *Card-order s*
shows $r \leq o\ cmax\ r\ s \wedge s \leq o\ cmax\ r\ s$
⟨*proof*⟩

lemmas *ordLeq-cmax1* = *ordLeq-cmax*[*THEN conjunct1*] **and**
ordLeq-cmax2 = *ordLeq-cmax*[*THEN conjunct2*]

lemma *finite-cmax*:
assumes *r*: *Card-order r* **and** *s*: *Card-order s*
shows *finite* (*Field* (*cmax r s*)) ⟷ *finite* (*Field r*) ∧ *finite* (*Field s*)
⟨*proof*⟩

end

12 Extending Well-founded Relations to Wellorders

theory *Wellorder-Extension*
imports *Main Order-Union*
begin

12.1 Extending Well-founded Relations to Wellorders

A *downset* (also lower set, decreasing set, initial segment, or downward closed set) is closed w.r.t. smaller elements.

definition *downset-on* **where**
downset-on A r = ($\forall x\ y.\ (x, y) \in r \wedge y \in A \longrightarrow x \in A$)

lemma *downset-onI*:
($\bigwedge x\ y.\ (x, y) \in r \Longrightarrow y \in A \Longrightarrow x \in A$) ⟹ *downset-on A r*
⟨*proof*⟩

lemma *downset-onD*:
downset-on A r ⟹ $(x, y) \in r \Longrightarrow y \in A \Longrightarrow x \in A$
⟨*proof*⟩

Extensions of relations w.r.t. a given set.

definition *extension-on* **where**
extension-on A r s = ($\forall x{\in}A.\ \forall y{\in}A.\ (x, y) \in s \longrightarrow (x, y) \in r$)

lemma *extension-onI*:
($\bigwedge x\ y.\ [\![x \in A; y \in A; (x, y) \in s]\!] \Longrightarrow (x, y) \in r$) ⟹ *extension-on A r s*
⟨*proof*⟩

lemma *extension-onD*:
  extension-on $A\ r\ s \Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow (x,\ y) \in s \Longrightarrow (x,\ y) \in r$
  ⟨*proof*⟩

lemma *downset-on-Union*:
  assumes $\bigwedge r.\ r \in R \Longrightarrow$ *downset-on* (*Field* r) p
  shows *downset-on* (*Field* ($\bigcup R$)) p
  ⟨*proof*⟩

lemma *chain-subset-extension-on-Union*:
  assumes $chain_{\subseteq}\ R$ **and** $\bigwedge r.\ r \in R \Longrightarrow$ *extension-on* (*Field* r) $r\ p$
  shows *extension-on* (*Field* ($\bigcup R$)) ($\bigcup R$) p
  ⟨*proof*⟩

lemma *downset-on-empty* [*simp*]: *downset-on* {} p
  ⟨*proof*⟩

lemma *extension-on-empty* [*simp*]: *extension-on* {} $p\ q$
  ⟨*proof*⟩

Every well-founded relation can be extended to a wellorder.

theorem *well-order-extension*:
  assumes *wf* p
  shows $\exists w.\ p \subseteq w \land$ *Well-order* w
⟨*proof*⟩

Every well-founded relation can be extended to a total wellorder.

corollary *total-well-order-extension*:
  assumes *wf* p
  shows $\exists w.\ p \subseteq w \land$ *Well-order* $w \land$ *Field* w = *UNIV*
⟨*proof*⟩

corollary *well-order-on-extension*:
  assumes *wf* p **and** *Field* $p \subseteq A$
  shows $\exists w.\ p \subseteq w \land$ *well-order-on* $A\ w$
⟨*proof*⟩

end

13 Theory of Ordinals and Cardinals

theory *Cardinals*
imports *Ordinal-Arithmetic Cardinal-Arithmetic Wellorder-Extension*
begin

end

14 Sets Strictly Bounded by an Infinite Cardinal

theory *Bounded-Set*
imports *Cardinals*
begin

typedef $('a, 'k)$ *bset* (- *set*[-] [*22*, *21*] *21*) =
  $\{A :: 'a\ set.\ |A| <o\ natLeq +c\ |UNIV :: 'k\ set|\}$
  morphisms *set-bset Abs-bset*
  ⟨*proof*⟩

setup-lifting *type-definition-bset*

lift-definition *map-bset* ::
  $('a \Rightarrow 'b) \Rightarrow 'a\ set['k] \Rightarrow 'b\ set['k]$ **is** *image*
  ⟨*proof*⟩

lift-definition *rel-bset* ::
  $('a \Rightarrow 'b \Rightarrow bool) \Rightarrow 'a\ set['k] \Rightarrow 'b\ set['k] \Rightarrow bool$ **is** *rel-set*
  ⟨*proof*⟩

lift-definition *bempty* :: $'a\ set['k]$ **is** $\{\}$
  ⟨*proof*⟩

lift-definition *binsert* :: $'a \Rightarrow 'a\ set['k] \Rightarrow 'a\ set['k]$ **is** *insert*
  ⟨*proof*⟩

definition *bsingleton* **where**
  bsingleton x = binsert x bempty

lemma *set-bset-to-set-bset*: $|A| <o\ natLeq +c\ |UNIV :: 'k\ set| \Longrightarrow$
  set-bset (*the-inv set-bset A* :: $'a\ set['k]$) = *A*
  ⟨*proof*⟩

lemma *rel-bset-aux-infinite*:
  fixes $a :: 'a\ set['k]$ **and** $b :: 'b\ set['k]$
  shows $(\forall t \in set\text{-}bset\ a.\ \exists u \in set\text{-}bset\ b.\ R\ t\ u) \wedge (\forall u \in set\text{-}bset\ b.\ \exists t \in set\text{-}bset\ a.\ R\ t\ u) \longleftrightarrow$
   $((BNF\text{-}Def.Grp\ \{a.\ set\text{-}bset\ a \subseteq \{(a, b).\ R\ a\ b\}\}\ (map\text{-}bset\ fst))^{-1-1}\ OO$
    $BNF\text{-}Def.Grp\ \{a.\ set\text{-}bset\ a \subseteq \{(a, b).\ R\ a\ b\}\}\ (map\text{-}bset\ snd))\ a\ b$ (**is** $?L \longleftrightarrow ?R$)
⟨*proof*⟩

bnf $'a\ set['k]$
  map: *map-bset*
  sets: *set-bset*
  bd: *natLeq +c card-suc* $|UNIV :: 'k\ set|$
  wits: *bempty*
  rel: *rel-bset*

$\langle proof \rangle$

lemma *map-bset-bempty*[*simp*]: *map-bset f bempty* = *bempty*
$\langle proof \rangle$

lemma *map-bset-binsert*[*simp*]: *map-bset f* (*binsert x X*) = *binsert* (*f x*) (*map-bset f X*)
$\langle proof \rangle$

lemma *map-bset-bsingleton*: *map-bset f* (*bsingleton x*) = *bsingleton* (*f x*)
$\langle proof \rangle$

lemma *bempty-not-binsert*: *bempty* $\neq$ *binsert x X binsert x X* $\neq$ *bempty*
$\langle proof \rangle$

lemma *bempty-not-bsingleton*[*simp*]: *bempty* $\neq$ *bsingleton x bsingleton x* $\neq$ *bempty*
$\langle proof \rangle$

lemma *bsingleton-inj*[*simp*]: *bsingleton x* = *bsingleton y* $\longleftrightarrow$ *x* = *y*
$\langle proof \rangle$

lemma *rel-bsingleton*[*simp*]:
rel-bset R (*bsingleton x1*) (*bsingleton x2*) = *R x1 x2*
$\langle proof \rangle$

lemma *rel-bset-bsingleton*[*simp*]:
rel-bset R (*bsingleton x1*) = (λX. *X* $\neq$ *bempty* $\wedge$ ($\forall$ *x2*$\in$*set-bset X*. *R x1 x2*))
rel-bset R X (*bsingleton x2*) = (*X* $\neq$ *bempty* $\wedge$ ($\forall$ *x1*$\in$*set-bset X*. *R x1 x2*))
$\langle proof \rangle$

lemma *rel-bset-bempty*[*simp*]:
rel-bset R bempty X = (*X* = *bempty*)
rel-bset R Y bempty = (*Y* = *bempty*)
$\langle proof \rangle$

definition *bset-of-option* **where**
bset-of-option = *case-option bempty bsingleton*

lift-definition *bgraph* :: ($'a \Rightarrow$ $'b$ *option*) $\Rightarrow$ ($'a \times$ $'b$) *set*[$'a$ *set*] **is**
λf. {(*a*, *b*). *f a* = *Some b*}
$\langle proof \rangle$

lemma *rel-bset-False*[*simp*]: *rel-bset* (λx *y*. *False*) *x y* = (*x* = *bempty* $\wedge$ *y* = *bempty*)
$\langle proof \rangle$

lemma *rel-bset-of-option*[*simp*]:
rel-bset R (*bset-of-option x1*) (*bset-of-option x2*) = *rel-option R x1 x2*
$\langle proof \rangle$

lemma *rel-bgraph*[*simp*]:
rel-bset (*rel-prod* (=) *R*) (*bgraph f1*) (*bgraph f2*) = *rel-fun* (=) (*rel-option R*) *f1* *f2*
⟨*proof*⟩

lemma *set-bset-bsingleton*[*simp*]:
set-bset (*bsingleton x*) = {*x*}
⟨*proof*⟩

lemma *binsert-absorb*[*simp*]: *binsert a* (*binsert a x*) = *binsert a x*
⟨*proof*⟩

lemma *map-bset-eq-bempty-iff*[*simp*]: *map-bset f X* = *bempty* ⟷ *X* = *bempty*
⟨*proof*⟩

lemma *map-bset-eq-bsingleton-iff*[*simp*]:
map-bset f X = *bsingleton x* ⟷ (*set-bset X* ≠ {} ∧ (∀ *y* ∈ *set-bset X*. *f y* = *x*))
⟨*proof*⟩

lift-definition *bCollect* :: (′*a* ⇒ *bool*) ⇒ ′*a set*[′*a set*] **is** *Collect*
⟨*proof*⟩

lift-definition *bmember* :: ′*a* ⇒ ′*a set*[′*k*] ⇒ *bool* **is** (∈) ⟨*proof*⟩

lemma *bmember-bCollect*[*simp*]: *bmember a* (*bCollect P*) = *P a*
⟨*proof*⟩

lemma *bset-eq-iff*: *A* = *B* ⟷ (∀ *a*. *bmember a A* = *bmember a B*)
⟨*proof*⟩

locale *bset-lifting*
begin

declare *bset.rel-eq*[*relator-eq*]
declare *bset.rel-mono*[*relator-mono*]
declare *bset.rel-compp*[*symmetric*, *relator-distr*]
declare *bset.rel-transfer*[*transfer-rule*]

lemma *bset-quot-map*[*quot-map*]: *Quotient R Abs Rep T* ⟹
Quotient (*rel-bset R*) (*map-bset Abs*) (*map-bset Rep*) (*rel-bset T*)
⟨*proof*⟩

lemma *set-relator-eq-onp* [*relator-eq-onp*]:
rel-bset (*eq-onp P*) = *eq-onp* (λ*A*. *Ball* (*set-bset A*) *P*)
⟨*proof*⟩

end

end