

Ordinals and cardinals in HOL

Andrei Popescu & Dmitriy Traytel

Contents

1	Introduction	3
2	More on Injections, Bijections and Inverses	5
2.1	Purely functional properties	5
2.2	Properties involving finite and infinite sets	6
2.3	Properties involving Hilbert choice	6
2.4	Other facts	6
3	Basics on Order-Like Relations	7
3.1	The upper and lower bounds operators	7
3.2	Properties depending on more than one relation	12
4	More on Well-Founded Relations	12
4.1	Well-founded recursion via genuine fixpoints	12
5	Well-Order Relations	13
5.1	Auxiliaries	13
5.2	Well-founded induction and recursion adapted to non-strict well-order relations	13
5.2.1	Properties of <code>max2</code>	13
5.2.2	Properties of <code>minim</code>	14
5.2.3	Properties of <code>supr</code>	14
5.2.4	Properties of <code>successor</code>	15
5.2.5	Properties of <code>order filters</code>	16
5.2.6	Other properties	16
6	Well-Order Embeddings	17
6.1	Auxiliaries	17
6.2	(Well-order) embeddings, strict embeddings, isomorphisms and order-compatible functions	17
6.3	Uniqueness of embeddings	18
7	Order Union	19

8	Constructions on Wellorders	21
8.1	Order filters versus restrictions and embeddings	21
8.2	Ordering the well-orders by existence of embeddings	21
8.3	Copy via direct images	23
8.4	The maxim among a finite set of ordinals	23
8.5	Limit and sucesor ordinals	25
8.5.1	Successor and limit elements of an ordinal	26
8.5.2	Well-order recursion with (zero), sucesor, and limit	29
8.5.3	Well-order succ-lim induction	30
8.6	Projections of wellorders	31
9	Ordinal Arithmetic	32
10	Cardinal-Order Relations	45
10.1	Cardinal of a set	46
10.2	Cardinals versus set operations on arbitrary sets	47
10.3	Cardinals versus set operations involving infinite sets	53
10.4	Cardinals versus lists	53
10.5	Cardinals versus the finite powerset operator	55
10.6	The cardinal ω and the finite cardinals	56
10.6.1	First as well-orders	56
10.6.2	Then as cardinals	57
10.6.3	"Backward compatibility" with the numeric cardinal operator for finite sets	58
10.7	The successor of a cardinal	59
10.8	Others	59
10.9	Infinite cardinals are limit ordinals	62
10.10	Regular vs. stable cardinals	63
11	Cardinal Arithmetic	63
11.1	Binary sum	64
11.2	Product	64
11.3	Exponentiation	64
11.4	Powerset	66
11.5	Inverse image	67
11.6	Maximum	67
12	Extending Well-founded Relations to Wellorders	68
12.1	Extending Well-founded Relations to Wellorders	68
13	Theory of Ordinals and Cardinals	69

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-order 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:

¹The “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 sets³ 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.

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

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

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

⁵This 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 ∈ A
```

shows $\text{bij-betw } f (A - \{a\}) (A' - \{f a\})$
<proof>

2.2 Properties involving finite and infinite sets

lemma *bij-betw-inv-into-RIGHT*:
assumes $\text{BIJ: } \text{bij-betw } f A A'$ **and** $\text{SUB: } B' \leq A'$
shows $f '(inv-into A f) 'B' = B'$
<proof>

lemma *bij-betw-inv-into-RIGHT-LEFT*:
assumes $\text{BIJ: } \text{bij-betw } f A A'$ **and** $\text{SUB: } B' \leq A'$ **and**
 $\text{IM: } (inv-into A f) 'B' = B$
shows $f 'B = B'$
<proof>

lemma *bij-betw-inv-into-twice*:
assumes $\text{bij-betw } f A A'$
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 $\text{BIJ: } \text{bij-betw } f A A'$ **and** $\text{SUB: } B \leq A$
shows $(inv-into A f) '(f 'B) = B$
<proof>

lemma *bij-betw-inv-into-LEFT-RIGHT*:
assumes $\text{BIJ: } \text{bij-betw } f A A'$ **and** $\text{SUB: } B \leq A$ **and**
 $\text{IM: } f 'B = B'$
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*:
 $\text{bij-betw } f \{0 ..< m::nat\} \{0 ..< n\} \implies m = n$
<proof>

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

<proof>

lemma *atLeastLessThan-less-eq2*:

assumes *inj-on* $f \{0..<(m::nat)\} f ' \{0..<m\} \leq \{0..<n\}$

shows $m \leq n$

<proof>

lemma *atLeastLessThan-less*:

$(\{0..<m\} < \{0..<n\}) = ((m::nat) < n)$

<proof>

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*: $aboveS \ r \ a \leq above \ r \ a$

<proof>

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

<proof>

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

<proof>

lemma *aboveS-notIn*: $a \notin aboveS \ r \ a$

<proof>

lemma *Refl-above-in*: $[[Refl \ r; a \in Field \ r]] \implies a \in above \ r \ a$

<proof>

lemma *in-Above-under*: $a \in Field \ r \implies a \in Above \ r \ (under \ r \ a)$

<proof>

lemma *in-Under-above*: $a \in Field \ r \implies a \in Under \ r \ (above \ r \ a)$

<proof>

lemma *in-UnderS-aboveS*: $a \in Field \ r \implies a \in UnderS \ r \ (aboveS \ r \ a)$

<proof>

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

<proof>

lemma *subset-Above-Under*: $B \leq \text{Field } r \implies B \leq \text{Above } r (\text{Under } r B)$
<proof>

lemma *subset-Under-Above*: $B \leq \text{Field } r \implies B \leq \text{Under } r (\text{Above } r B)$
<proof>

lemma *subset-AboveS-UnderS*: $B \leq \text{Field } r \implies B \leq \text{AboveS } r (\text{UnderS } r B)$
<proof>

lemma *subset-UnderS-AboveS*: $B \leq \text{Field } r \implies B \leq \text{UnderS } r (\text{AboveS } r B)$
<proof>

lemma *Under-Above-Galois*:
 $\llbracket B \leq \text{Field } r; C \leq \text{Field } r \rrbracket \implies (B \leq \text{Above } r C) = (C \leq \text{Under } r B)$
<proof>

lemma *UnderS-AboveS-Galois*:
 $\llbracket B \leq \text{Field } r; C \leq \text{Field } r \rrbracket \implies (B \leq \text{AboveS } r C) = (C \leq \text{UnderS } r B)$
<proof>

lemma *Refl-above-aboveS*:
assumes *REFL*: *Refl* r **and** *IN*: $a \in \text{Field } r$
shows $\text{above } r a = \text{aboveS } r a \cup \{a\}$
<proof>

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

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

lemma *under-empty*: $a \notin \text{Field } r \implies \text{under } r a = \{\}$
<proof>

lemma *Under-Field*: $\text{Under } r A \leq \text{Field } r$
<proof>

lemma *UnderS-Field*: $\text{UnderS } r A \leq \text{Field } r$
<proof>

lemma *above-Field*: $\text{above } r a \leq \text{Field } r$
<proof>

lemma *aboveS-Field*: $\text{aboveS } r a \leq \text{Field } r$
<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*: $A \leq B \implies Under\ r\ B \leq Under\ r\ A$
<proof>

lemma *UnderS-decr*: $A \leq B \implies UnderS\ r\ B \leq UnderS\ r\ A$
<proof>

lemma *Above-decr*: $A \leq B \implies Above\ r\ B \leq Above\ r\ A$
<proof>

lemma *AboveS-decr*: $A \leq B \implies AboveS\ r\ B \leq AboveS\ r\ A$
<proof>

lemma *under-incl-iff*:

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

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

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

lemma *under-trans*:
assumes *TRANS*: *trans r* **and**
IN1: $a \in \text{under } r \ b$ **and** *IN2*: $b \in \text{under } r \ c$
shows $a \in \text{under } r \ c$
(*proof*)

lemma *under-underS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in \text{under } r \ b$ **and** *IN2*: $b \in \text{underS } r \ c$
shows $a \in \text{underS } r \ c$
(*proof*)

lemma *underS-under-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in \text{underS } r \ b$ **and** *IN2*: $b \in \text{under } r \ c$
shows $a \in \text{underS } r \ c$
(*proof*)

lemma *underS-underS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in \text{underS } r \ b$ **and** *IN2*: $b \in \text{underS } r \ c$
shows $a \in \text{underS } r \ c$
(*proof*)

lemma *above-trans*:
assumes *TRANS*: *trans r* **and**
IN1: $b \in \text{above } r \ a$ **and** *IN2*: $c \in \text{above } r \ b$
shows $c \in \text{above } r \ a$
(*proof*)

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

<proof>

lemma *aboveS-above-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $b \in \text{aboveS } r \ a$ and IN2: $c \in \text{above } r \ b$

shows *$c \in \text{aboveS } r \ a$*

<proof>

lemma *aboveS-aboveS-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $b \in \text{aboveS } r \ a$ and IN2: $c \in \text{aboveS } r \ b$

shows *$c \in \text{aboveS } r \ a$*

<proof>

lemma *under-Under-trans:*

assumes *TRANS: trans r and*

IN1: $a \in \text{under } r \ b$ and IN2: $b \in \text{Under } r \ C$

shows *$a \in \text{Under } r \ C$*

<proof>

lemma *underS-Under-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $a \in \text{underS } r \ b$ and IN2: $b \in \text{Under } r \ C$

shows *$a \in \text{UnderS } r \ C$*

<proof>

lemma *underS-UnderS-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $a \in \text{underS } r \ b$ and IN2: $b \in \text{UnderS } r \ C$

shows *$a \in \text{UnderS } r \ C$*

<proof>

lemma *above-Above-trans:*

assumes *TRANS: trans r and*

IN1: $a \in \text{above } r \ b$ and IN2: $b \in \text{Above } r \ C$

shows *$a \in \text{Above } r \ C$*

<proof>

lemma *aboveS-Above-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $a \in \text{aboveS } r \ b$ and IN2: $b \in \text{Above } r \ C$

shows *$a \in \text{AboveS } r \ C$*

<proof>

lemma *above-AboveS-trans:*

assumes *TRANS: trans r and ANTISYM: antisym r and*

IN1: $a \in \text{above } r \ b$ and IN2: $b \in \text{AboveS } r \ C$

shows *$a \in \text{AboveS } r \ C$*

<proof>

lemma *aboveS-AboveS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in \text{aboveS } r \ b$ **and** *IN2*: $b \in \text{AboveS } r \ C$
shows $a \in \text{AboveS } r \ C$
 $\langle \text{proof} \rangle$

lemma *under-UnderS-trans*:
assumes *TRANS*: *trans r* **and** *ANTISYM*: *antisym r* **and**
IN1: $a \in \text{under } r \ b$ **and** *IN2*: $b \in \text{UnderS } r \ C$
shows $a \in \text{UnderS } r \ C$
 $\langle \text{proof} \rangle$

3.2 Properties depending on more than one relation

lemma *under-incr2*:
 $r \leq r' \implies \text{under } r \ a \leq \text{under } r' \ a$
 $\langle \text{proof} \rangle$

lemma *underS-incr2*:
 $r \leq r' \implies \text{underS } r \ a \leq \text{underS } r' \ a$
 $\langle \text{proof} \rangle$

lemma *above-incr2*:
 $r \leq r' \implies \text{above } r \ a \leq \text{above } r' \ a$
 $\langle \text{proof} \rangle$

lemma *aboveS-incr2*:
 $r \leq r' \implies \text{aboveS } r \ a \leq \text{aboveS } r' \ a$
 $\langle \text{proof} \rangle$

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 $r :: ('a * 'a) \text{ set}$ **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*: $adm\text{-}wf\ r\ H$ **and** *fFP*: $f = H\ f$ **and** *gFP*: $g = H\ g$
shows $f = g$
 $\langle proof \rangle$

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*: $adm\text{-}wf\ r\ H$ **and**
 fp : $f = H\ f$
shows $f = wfrec\ r\ H$
 $\langle proof \rangle$

end

5 Well-Order Relations

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

context *wo-rel*
begin

5.1 Auxiliaries

lemma *PREORD*: *Preorder* r
 $\langle proof \rangle$

lemma *PARORD*: *Partial-order* r
 $\langle proof \rangle$

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

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

lemma *worec-unique-fixpoint*:
assumes *ADM*: $adm\text{-}wo\ H$ **and** fp : $f = H\ f$
shows $f = worec\ H$
 $\langle proof \rangle$

5.2.1 Properties of max2

lemma *max2-iff*:

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

5.2.2 Properties of minim

lemma *minim-Under*:
 $\llbracket B \leq \text{Field } r; B \neq \{\} \rrbracket \implies \text{minim } B \in \text{Under } B$
 $\langle \text{proof} \rangle$

lemma *equals-minim-Under*:
 $\llbracket B \leq \text{Field } r; a \in B; a \in \text{Under } B \rrbracket$
 $\implies a = \text{minim } B$
 $\langle \text{proof} \rangle$

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

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

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

5.2.3 Properties of supr

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

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

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

lemma *supr-least*:
 $\llbracket B \leq \text{Field } r; a \in \text{Field } r; (\bigwedge b. b \in B \implies (b, a) \in r) \rrbracket$
 $\implies (\text{supr } B, a) \in r$

<proof>

lemma *equals-supr-Above:*

assumes $a \in \text{Above } B \wedge a'. a' \in \text{Above } B \implies (a, a') \in r$

shows $a = \text{supr } B$

<proof>

lemma *equals-supr:*

assumes $SUB: B \leq \text{Field } r$ **and** $IN: a \in \text{Field } r$ **and**

$ABV: \bigwedge b. b \in B \implies (b, a) \in r$ **and**

$MINIM: \bigwedge a'. \llbracket a' \in \text{Field } r; \bigwedge b. b \in B \implies (b, a') \in r \rrbracket \implies (a, a') \in r$

shows $a = \text{supr } B$

<proof>

lemma *supr-inField:*

assumes $\text{Above } B \neq \{\}$

shows $\text{supr } B \in \text{Field } r$

<proof>

lemma *supr-above-Above:*

assumes $SUB: B \leq \text{Field } r$ **and** $ABOVE: \text{Above } B \neq \{\}$

shows $\text{Above } B = \text{above } (\text{supr } B)$

<proof>

lemma *supr-under:*

assumes $a \in \text{Field } r$

shows $a = \text{supr } (\text{under } a)$

<proof>

5.2.4 Properties of successor

lemma *suc-least:*

$\llbracket B \leq \text{Field } r; a \in \text{Field } r; (\bigwedge b. b \in B \implies a \neq b \wedge (b, a) \in r) \rrbracket$

$\implies (\text{suc } B, a) \in r$

<proof>

lemma *equals-suc:*

assumes $SUB: B \leq \text{Field } r$ **and** $IN: a \in \text{Field } r$ **and**

$ABVS: \bigwedge b. b \in B \implies a \neq b \wedge (b, a) \in r$ **and**

$MINIM: \bigwedge a'. \llbracket a' \in \text{Field } r; \bigwedge b. b \in B \implies a' \neq b \wedge (b, a') \in r \rrbracket \implies (a, a') \in r$

r

shows $a = \text{suc } B$

<proof>

lemma *suc-above-AboveS:*

assumes $SUB: B \leq \text{Field } r$ **and**

$ABOVE: \text{AboveS } B \neq \{\}$

shows $\text{AboveS } B = \text{above } (\text{suc } B)$

<proof>

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

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

5.2.5 Properties of order filters

lemma *ofilter-Under[simp]*:
assumes $A \leq \text{Field } r$
shows $\text{ofilter}(\text{Under } A)$
 $\langle \text{proof} \rangle$

lemma *ofilter-UnderS[simp]*:
assumes $A \leq \text{Field } r$
shows $\text{ofilter}(\text{UnderS } A)$
 $\langle \text{proof} \rangle$

lemma *ofilter-Int[simp]*: $\llbracket \text{ofilter } A; \text{ofilter } B \rrbracket \implies \text{ofilter}(A \text{ Int } B)$
 $\langle \text{proof} \rangle$

lemma *ofilter-Un[simp]*: $\llbracket \text{ofilter } A; \text{ofilter } B \rrbracket \implies \text{ofilter}(A \cup B)$
 $\langle \text{proof} \rangle$

lemma *ofilter-INTER*:
 $\llbracket I \neq \{\}; \bigwedge i. i \in I \implies \text{ofilter}(A \ i) \rrbracket \implies \text{ofilter}(\bigcap i \in I. A \ i)$
 $\langle \text{proof} \rangle$

lemma *ofilter-Inter*:
 $\llbracket S \neq \{\}; \bigwedge A. A \in S \implies \text{ofilter } A \rrbracket \implies \text{ofilter}(\bigcap S)$
 $\langle \text{proof} \rangle$

lemma *ofilter-Union*:
 $(\bigwedge A. A \in S \implies \text{ofilter } A) \implies \text{ofilter}(\bigcup S)$
 $\langle \text{proof} \rangle$

lemma *ofilter-under-Union*:
 $\text{ofilter } A \implies A = \bigcup \{\text{under } a \mid a. a \in A\}$
 $\langle \text{proof} \rangle$

5.2.6 Other properties

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

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* $\equiv wo-rel.worec$

abbreviation *adm-wo* $\equiv wo-rel.adm-wo$

abbreviation *isMinim* $\equiv wo-rel.isMinim$

abbreviation *minim* $\equiv wo-rel.minim$

abbreviation *max2* $\equiv wo-rel.max2$

abbreviation *supr* $\equiv wo-rel.supr$

abbreviation *suc* $\equiv wo-rel.suc$

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: $\bigwedge i. i \in I \implies ofilter\ r\ (A\ i)$ **and**

BIJ: $\bigwedge i. i \in I \implies bij-betw\ f\ (A\ i)\ (A'\ i)$

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 $\bigwedge a. a \in Field\ r \implies f\ a = g\ a$ **and** *embed* $r\ r'\ f$

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

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

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

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

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

lemma $comp-compat$:
 $\llbracket compat\ r\ r'\ f; compat\ r'\ r''\ f \rrbracket \implies compat\ r\ r''\ (f' o f)$
 $\langle proof \rangle$

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

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

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 $iso\ r'\ r\ (inv-into\ (Field\ r)\ f)$
<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*:

$\llbracket \text{Field } r \text{ Int Field } r' = \{\}; \text{Preorder } r; \text{Preorder } r' \rrbracket \implies \text{Preorder } (r \text{ Osum } r')$
(*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*:

$\llbracket \text{Field } r \text{ Int Field } r' = \{\}; \text{Partial-order } r; \text{Partial-order } r' \rrbracket \implies$
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*:

$\llbracket \text{Field } r \text{ Int Field } r' = \{\}; \text{Linear-order } r; \text{Linear-order } r' \rrbracket \implies \text{Linear-order } (r$
Osum *r'*)
(*proof*)

lemma *Osum-minus-Id1*:

assumes $r \leq \text{Id}$
shows $(r \text{ Osum } r') - \text{Id} \leq (r' - \text{Id}) \cup (\text{Field } r \times \text{Field } r')$
(*proof*)

lemma *Osum-minus-Id2*:

assumes $r' \leq \text{Id}$
shows $(r \text{ Osum } r') - \text{Id} \leq (r - \text{Id}) \cup (\text{Field } r \times \text{Field } r')$
(*proof*)

lemma *Osum-minus-Id*:

assumes *TOT*: *Total* *r* **and** *TOT'*: *Total* *r'* **and**
NID: $\neg (r \leq \text{Id})$ **and** *NID'*: $\neg (r' \leq \text{Id})$
shows $(r \text{ Osum } r') - \text{Id} \leq (r - \text{Id}) \text{ Osum } (r' - \text{Id})$
(*proof*)

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

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 $wf((r \text{ Osum } r') - Id)$
 $\langle proof \rangle$

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

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 $(A = B) = iso (Restr\ r\ A) (Restr\ r\ B) id$
 $\langle proof \rangle$

8.2 Ordering the well-orders by existence of embeddings

corollary *ordLeq-refl-on*: $refl\ on\ \{r.\ Well\ order\ r\}$ $ordLeq$
 $\langle proof \rangle$

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

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

lemma *ordIso-Well-order-simp[simp]*:
assumes $r =_o r'$
shows $Well\text{-}order\ r \wedge Well\text{-}order\ r'$
⟨*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 ≤_o r*
 ⟨*proof*⟩

corollary *under-Restr-ordLeq*:
Well-order r \implies *Restr r (under r a) ≤_o r*
 ⟨*proof*⟩

8.3 Copy via direct images

lemma *Id-dir-image*: *dir-image Id f ≤ Id*
 ⟨*proof*⟩

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

lemma *Int-dir-image*:
assumes *inj-on f (Field r1 ∪ 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 ≤_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 ≤ B*
shows $\exists r'. \text{well-order-on } B \ r' \wedge r \leq_o r'$
 ⟨*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 $r \leq_o\ p$

$\langle proof \rangle$

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 successor 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 $\bigwedge a. (\bigwedge a1. a1 \in underS\ a \implies P\ a1) \implies P\ a$
shows $P\ a$
 ⟨proof⟩

lemma *suc-underS*:

assumes $B: B \subseteq \text{Field } r$ **and** $A: \text{AboveS } B \neq \{\}$ **and** $b: b \in B$
shows $b \in \text{underS } (\text{suc } B)$
<proof>

lemma *underS-supr*:

assumes $bA: b \in \text{underS } (\text{supr } A)$ **and** $A: A \subseteq \text{Field } r$
shows $\exists a \in A. b \in \text{underS } a$
<proof>

lemma *underS-suc*:

assumes $bA: b \in \text{underS } (\text{suc } A)$ **and** $A: A \subseteq \text{Field } r$
shows $\exists a \in A. b \in \text{under } a$
<proof>

lemma (*in wo-rel*) *in-underS-supr*:

assumes $j \in \text{underS } i$ **and** $i \in A$ **and** $A \subseteq \text{Field } r$ **and** $\text{Above } A \neq \{\}$
shows $j \in \text{underS } (\text{supr } A)$
<proof>

lemma *inj-on-Field*:

assumes $A: A \subseteq \text{Field } r$ **and** $f: \bigwedge a b. \llbracket a \in A; b \in A; a \in \text{underS } b \rrbracket \implies f a \neq f b$
shows *inj-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 $\bigwedge j1 j2. \llbracket j2 \in \text{underS } i; (j1, j2) \in r \rrbracket \implies R j1$ *initial-segment-of* $R j2$
shows $\{R j \mid j. j \in \text{underS } i\} \in \text{Chains init-seg-of}$
<proof>

lemma (*in wo-rel*) *Field-init-seg-of-Collect*:

assumes $\bigwedge j1 j2. \llbracket j2 \in \text{Field } r; (j1, j2) \in r \rrbracket \implies R j1$ *initial-segment-of* $R j2$
shows $\{R j \mid j. j \in \text{Field } r\} \in \text{Chains init-seg-of}$
<proof>

8.5.1 Successor and limit elements of an ordinal

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

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

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

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

lemma $zero\text{-}smallest[simp]$:
assumes $j \in Field\ r$ shows $(zero, j) \in r$
 $\langle proof \rangle$

lemma $zero\text{-}in\text{-}Field$: assumes $Field\ r \neq \{\}$ shows $zero \in Field\ r$
 $\langle proof \rangle$

lemma $leq\text{-}zero\text{-}imp[simp]$:
 $(x, zero) \in r \implies x = zero$
 $\langle proof \rangle$

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

lemma $under\text{-}zero[simp]$:
assumes $Field\ r \neq \{\}$ shows $under\ zero = \{zero\}$
 $\langle proof \rangle$

lemma $underS\text{-}zero[simp,intro]$: $underS\ zero = \{\}$
 $\langle proof \rangle$

lemma $isSucc\text{-}succ$: $aboveS\ i \neq \{\} \implies isSucc\ (succ\ i)$
 $\langle proof \rangle$

lemma $succ\text{-}in\text{-}diff$:
assumes $aboveS\ i \neq \{\}$ shows $(i, succ\ i) \in r \wedge succ\ i \neq i$
 $\langle proof \rangle$

lemmas $succ\text{-}in[simp] = succ\text{-}in\text{-}diff[THEN\ conjunct1]$
lemmas $succ\text{-}diff[simp] = succ\text{-}in\text{-}diff[THEN\ conjunct2]$

lemma $succ\text{-}in\text{-}Field[simp]$:
assumes $aboveS\ i \neq \{\}$ shows $succ\ i \in Field\ r$
 $\langle proof \rangle$

lemma $succ\text{-}not\text{-}in$:
assumes $aboveS\ i \neq \{\}$ shows $(succ\ i, i) \notin r$
 $\langle proof \rangle$

lemma $not\text{-}isSucc\text{-}zero$: $\neg\ isSucc\ zero$
 $\langle proof \rangle$

lemma $isLim\text{-}zero[simp]$: $isLim\ zero$
 $\langle proof \rangle$

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 $(j, succ\ i) \in r \longleftrightarrow (j, i) \in r \vee j = succ\ i$

<proof>

lemma *underS-succ[simp]*:

assumes $aboveS\ i \neq \{\}$

shows $underS\ (succ\ i) = under\ i$

<proof>

lemma *succ-mono*:

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

lemma *under-succ[simp]*:

assumes $\text{aboveS } i \neq \{\}$
shows $\text{under } (\text{succ } i) = \text{insert } (\text{succ } i) (\text{under } i)$
 $\langle \text{proof} \rangle$

definition *mergeSL* :: $('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$

where

$\text{mergeSL } S L f i \equiv \text{if } \text{isSucc } i \text{ then } S (\text{pred } i) (f (\text{pred } i)) \text{ else } L f i$

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

definition *worecSL* :: $('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$

where $\text{worecSL } S L \equiv \text{worec } (\text{mergeSL } S L)$

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

lemma *mergeSL: adm-woL* $L \Longrightarrow \text{adm-wo } (\text{mergeSL } S L)$

$\langle \text{proof} \rangle$

lemma *worec-fixpoint1*: $\text{adm-wo } H \Longrightarrow \text{worec } H i = H (\text{worec } H) i$

$\langle \text{proof} \rangle$

lemma *worecSL-isSucc*:

assumes $a: \text{adm-woL } L$ **and** $i: \text{isSucc } i$

shows $\text{worecSL } S L i = S (\text{pred } i) (\text{worecSL } S L (\text{pred } i))$

$\langle \text{proof} \rangle$

lemma *worecSL-succ*:

assumes $a: \text{adm-woL } L$ **and** $i: \text{aboveS } j \neq \{\}$

shows $\text{worecSL } S L (\text{succ } j) = S j (\text{worecSL } S L j)$

$\langle \text{proof} \rangle$

lemma *worecSL-isLim*:

assumes $a: \text{adm-woL } L$ **and** $i: \text{isLim } i$

shows $\text{worecSL } S L i = L (\text{worecSL } S L) i$

$\langle \text{proof} \rangle$

definition *worecZSL* :: $'b \Rightarrow ('a \Rightarrow 'b \Rightarrow 'b) \Rightarrow (('a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'b$

where $\text{worecZSL } Z S L \equiv \text{worecSL } S (\lambda f a. \text{if } a = \text{zero} \text{ then } Z \text{ else } L f a)$

lemma *worecZSL-zero*:

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

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

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

8.5.3 Well-order succ-lim induction

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

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

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

definition $\text{isSuccOrd} \equiv \exists j \in \text{Field } r. \forall i \in \text{Field } r. (i, j) \in r$

definition $\text{isLimOrd} \equiv \neg \text{isSuccOrd}$

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

lemma isLimOrd-aboveS :
assumes $l: \text{isLimOrd}$ **and** $i: i \in \text{Field } r$
shows $\text{aboveS } i \neq \{\}$
 $\langle \text{proof} \rangle$

lemma $\text{succ-aboveS-isLimOrd}$:
assumes $\forall i \in \text{Field } r. \text{aboveS } i \neq \{\} \wedge \text{succ } i \in \text{Field } r$

shows *isLimOrd*
<proof>

lemma *isLim-iff*:
assumes *l: isLim i* **and** *j: j ∈ underS i*
shows $\exists k. k \in \text{underS } i \wedge j \in \text{underS } k$
<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') ∈ r*
shows *(f a, f a') ∈ s*
<proof>

lemma *oproj-Field*:
assumes *f: oproj r s f* **and** *a: a ∈ Field r*
shows *f a ∈ Field s*
<proof>

lemma *oproj-Field2*:
assumes *f: oproj r s f* **and** *a: b ∈ Field s*
shows $\exists a \in \text{Field } r. f a = b$
<proof>

lemma *oproj-under*:
assumes *f: oproj r s f* **and** *a: a ∈ under r a'*
shows *f a ∈ under s (f a')*
<proof>

theorem *embedI*:
assumes *r: Well-order r* **and** *s: Well-order s*
and *f: $\bigwedge a. a \in \text{Field } r \implies f a \in \text{Field } s \wedge f ' \text{underS } r a \subseteq \text{underS } s (f a)$*

shows $\exists g. \text{embed } r \text{ } s \text{ } g$
 $\langle \text{proof} \rangle$

corollary *ordLeq-def2*:

$r \leq_o s \iff \text{Well-order } r \wedge \text{Well-order } s \wedge$
 $(\exists f. \forall a \in \text{Field } r. f \ a \in \text{Field } s \wedge f \ ' \ \text{underS } r \ a \subseteq \text{underS } s \ (f \ a))$
 $\langle \text{proof} \rangle$

lemma *iso-oproj*:

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

theorem *oproj-embed*:

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

corollary *oproj-ordLeq*:

assumes $r: \text{Well-order } r$ **and** $s: \text{Well-order } s$ **and** $f: \text{oproj } r \text{ } s \text{ } f$
shows $s \leq_o r$
 $\langle \text{proof} \rangle$

end

9 Ordinal Arithmetic

theory *Ordinal-Arithmetic*

imports *Wellorder-Constructions*

begin

definition *osum* $:: 'a \text{ rel} \Rightarrow 'b \text{ rel} \Rightarrow ('a + 'b) \text{ rel}$ (**infixr** $+_o$ 70)

where

$r +_o r' = \text{map-prod } \text{Inl } \text{Inl} \ ' \ r \cup \text{map-prod } \text{Inr } \text{Inr} \ ' \ r' \cup$
 $\{(\text{Inl } a, \text{Inr } a') \mid a \ a'. a \in \text{Field } r \wedge a' \in \text{Field } r'\}$

lemma *Field-osum*: $\text{Field}(r +_o r') = \text{Inl} \ ' \ \text{Field } r \cup \text{Inr} \ ' \ \text{Field } r'$

$\langle \text{proof} \rangle$

lemma *osum-Refl*: $\llbracket \text{Refl } r; \text{Refl } r' \rrbracket \Longrightarrow \text{Refl } (r +_o r')$

$\langle \text{proof} \rangle$

lemma *osum-trans*:

assumes *TRANS*: $\text{trans } r$ **and** *TRANS'*: $\text{trans } r'$

shows $\text{trans } (r +_o r')$

$\langle \text{proof} \rangle$

lemma *osum-Preorder*: $\llbracket \text{Preorder } r; \text{Preorder } r' \rrbracket \Longrightarrow \text{Preorder } (r +_o r')$

$\langle proof \rangle$

lemma *osum-antisym*: $\llbracket antisym\ r; antisym\ r' \rrbracket \implies antisym\ (r +_o r')$
 $\langle proof \rangle$

lemma *osum-Partial-order*: $\llbracket Partial-order\ r; Partial-order\ r' \rrbracket \implies Partial-order\ (r +_o r')$
 $\langle proof \rangle$

lemma *osum-Total*: $\llbracket Total\ r; Total\ r' \rrbracket \implies Total\ (r +_o r')$
 $\langle proof \rangle$

lemma *osum-Linear-order*: $\llbracket Linear-order\ r; Linear-order\ r' \rrbracket \implies Linear-order\ (r +_o r')$
 $\langle proof \rangle$

lemma *osum-wf*:
assumes *WF*: *wf* *r* **and** *WF'*: *wf* *r'*
shows *wf* $(r +_o r')$
 $\langle proof \rangle$

lemma *osum-minus-Id*:
assumes *r*: *Total* *r* $\neg (r \leq Id)$ **and** *r'*: *Total* *r'* $\neg (r' \leq Id)$
shows $(r +_o r') - Id \leq (r - Id) +_o (r' - Id)$
 $\langle proof \rangle$

lemma *osum-minus-Id1*:
 $r \leq Id \implies (r +_o r') - Id \leq (Inl\ 'Field\ r \times Inr\ 'Field\ r') \cup (map-prod\ Inr\ Inr\ ' (r' - Id))$
 $\langle proof \rangle$

lemma *osum-minus-Id2*:
 $r' \leq Id \implies (r +_o r') - Id \leq (map-prod\ Inl\ Inl\ ' (r - Id)) \cup (Inl\ 'Field\ r \times Inr\ 'Field\ r')$
 $\langle proof \rangle$

lemma *osum-wf-Id*:
assumes *TOT*: *Total* *r* **and** *TOT'*: *Total* *r'* **and** *WF*: *wf* $(r - Id)$ **and** *WF'*: *wf* $(r' - Id)$
shows *wf* $((r +_o r') - Id)$
 $\langle proof \rangle$

lemma *osum-Well-order*:
assumes *WELL*: *Well-order* *r* **and** *WELL'*: *Well-order* *r'*
shows *Well-order* $(r +_o r')$
 $\langle proof \rangle$

lemma *osum-embedL*:
assumes *WELL*: *Well-order* *r* **and** *WELL'*: *Well-order* *r'*

shows $\text{embed } r (r +_o r')$ *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*: $\text{dir-image } r f = \text{map-prod } f f \text{ ' } r$
 ⟨*proof*⟩

lemma *map-prod-ordIso*: $\llbracket \text{Well-order } r; \text{inj-on } f \text{ (Field } r) \rrbracket \implies \text{map-prod } f f \text{ ' } r$
 $=_o r$
 ⟨*proof*⟩

definition *oprod* :: $'a \text{ rel} \Rightarrow 'b \text{ rel} \Rightarrow ('a \times 'b) \text{ rel}$ (**infixr** $*_o$ 80)

where $r *_o r' = \{((x1, y1), (x2, y2)).$
 $((y1, y2) \in r' - \text{Id} \wedge x1 \in \text{Field } r \wedge x2 \in \text{Field } r) \vee$
 $((y1, y2) \in \text{Restr Id (Field } r') \wedge (x1, x2) \in r)\}$

lemma *Field-oprod*: $\text{Field } (r *_o r') = \text{Field } r \times \text{Field } r'$
 ⟨*proof*⟩

lemma *oprod-Refl*: $\llbracket \text{Refl } r; \text{Refl } r' \rrbracket \implies \text{Refl } (r *_o r')$
 ⟨*proof*⟩

lemma *oprod-trans*:

assumes *trans r* *trans r'* *antisym r* *antisym r'*
shows *trans (r *_o r')*
 ⟨*proof*⟩

lemma *oprod-Preorder*: $\llbracket \text{Preorder } r; \text{Preorder } r'; \text{antisym } r; \text{antisym } r' \rrbracket \implies \text{Pre-order } (r *_o r')$
 ⟨*proof*⟩

lemma *oprod-antisym*: $\llbracket \text{antisym } r; \text{antisym } r' \rrbracket \implies \text{antisym } (r *_o r')$
 ⟨*proof*⟩

lemma *oprod-Partial-order*: $\llbracket \text{Partial-order } r; \text{Partial-order } r' \rrbracket \implies \text{Partial-order } (r *_o r')$
 ⟨*proof*⟩

lemma *oprod-Total*: $\llbracket \text{Total } r; \text{Total } r' \rrbracket \implies \text{Total } (r *_o r')$
 ⟨*proof*⟩

lemma *oprod-Linear-order*: $\llbracket \text{Linear-order } r; \text{Linear-order } r' \rrbracket \implies \text{Linear-order } (r *_o r')$
 ⟨*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* \implies *r* *o *r'* - *Id* \leq $\{((x,y1), (x,y2)). x \in \text{Field } r \wedge (y1, y2) \in (r' - \text{Id})\}$

<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 \implies *r* *o *r'* - *Id* \leq $\{((x1,y), (x2,y)). (x1, x2) \in (r - \text{Id}) \wedge y \in \text{Field } r'\}$

<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')* ($\lambda x. (x, \text{minim } r' (\text{Field } r'))$) (**is embed** - - ?f)

<proof>

corollary *oprod-ordLeq*: $\llbracket \text{Well-order } r; \text{Well-order } r'; r' \neq \{\} \rrbracket \implies r \leq_o r *o r'$

<proof>

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

lemma *support-Un[simp]*: $\text{support } z (A \cup B) f = \text{support } z A f \cup \text{support } z B f$
 ⟨proof⟩

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

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

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

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

lemma *finite-support*: $f \in \text{fin-support } z A \implies \text{finite } (\text{support } z A f)$
 ⟨proof⟩

lemma *fin-support-Field-osum*:
 $f \in \text{fin-support } z (\text{Inl } 'A \cup \text{Inr } 'B) \longleftrightarrow$
 $(f \circ \text{Inl}) \in \text{fin-support } z A \wedge (f \circ \text{Inr}) \in \text{fin-support } z B$ (**is** ?L \longleftrightarrow ?R1 \wedge ?R2)
 ⟨proof⟩

lemma *Func-upd*: $\llbracket f \in \text{Func } A B; x \in A; y \in B \rrbracket \implies f(x := y) \in \text{Func } A B$
 ⟨proof⟩

context *wo-rel*
begin

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

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

lemma *isMaxim-unique[intro]*: $\llbracket \text{isMaxim } A x; \text{isMaxim } A y \rrbracket \implies x = y$
 ⟨proof⟩

lemma *maxim-isMaxim*: $\llbracket \text{finite } A; A \neq \{\}; A \subseteq \text{Field } r \rrbracket \implies \text{isMaxim } A (\text{maxim } A)$
 ⟨proof⟩

lemma *maxim-in*: $\llbracket \text{finite } A; A \neq \{\}; A \subseteq \text{Field } r \rrbracket \implies \text{maxim } A \in A$
 ⟨proof⟩

lemma *maxim-greatest*: $\llbracket \text{finite } A; x \in A; A \subseteq \text{Field } r \rrbracket \implies (x, \text{maxim } A) \in r$

<proof>

lemma *isMaxim-zero*: $isMaxim\ A\ zero \implies A = \{zero\}$
<proof>

lemma *maxim-insert*:
assumes *finite A A ≠ {} A ⊆ Field r x ∈ Field r*
shows $maxim\ (insert\ x\ A) = max2\ x\ (maxim\ A)$
<proof>

lemma *maxim-Un*:
assumes *finite A A ≠ {} A ⊆ Field r finite B B ≠ {} B ⊆ Field r*
shows $maxim\ (A \cup B) = max2\ (maxim\ A)\ (maxim\ B)$
<proof>

lemma *maxim-insert-zero*:
assumes *finite A A ≠ {} A ⊆ Field r*
shows $maxim\ (insert\ zero\ A) = maxim\ A$
<proof>

lemma *maxim-equality*: $isMaxim\ A\ x \implies maxim\ A = x$
<proof>

lemma *maxim-singleton*:
 $x \in Field\ r \implies maxim\ \{x\} = x$
<proof>

lemma *maxim-Int*: $\llbracket finite\ A; A \neq \{\}; A \subseteq Field\ r; maxim\ A \in B \rrbracket \implies maxim\ (A \cap B) = maxim\ A$
<proof>

lemma *maxim-mono*: $\llbracket X \subseteq Y; finite\ Y; X \neq \{\}; Y \subseteq Field\ r \rrbracket \implies (maxim\ X, maxim\ Y) \in r$
<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*: $x \in Field\ r \implies zero \in under\ x$
<proof>

end

definition *FinFunc* $r\ s = Func\ (Field\ s)\ (Field\ r) \cap fin-support\ (zero\ r)\ (Field\ s)$

lemma *FinFuncD*: $\llbracket f \in FinFunc\ r\ s; x \in Field\ s \rrbracket \implies f\ x \in Field\ r$
<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* \equiv *support r.zero (Field s)*
abbreviation *FINFUNC* \equiv *FinFunc r s*
lemmas *FINFUNCD* = *FinFuncD[of - r s]*

lemma *fun-diff-alt*: $\{a \in \text{Field } s. f a \neq g a\} = (\text{SUPP } f \cup \text{SUPP } g) \cap \{a. f a \neq g a\}$
<proof>

lemma *max-fun-diff-alt*:
s.max-fun-diff f g = s.maxim ((SUPP f \cup SUPP g) \cap {a. f a \neq g a})
<proof>

lemma *isMaxim-max-fun-diff*: $\llbracket f \neq g; f \in \text{FINFUNC}; g \in \text{FINFUNC} \rrbracket \implies$
s.isMaxim {a \in Field s. f a \neq g a} (s.max-fun-diff f g)
<proof>

lemma *max-fun-diff-in*: $\llbracket f \neq g; f \in \text{FINFUNC}; g \in \text{FINFUNC} \rrbracket \implies$
s.max-fun-diff f g \in {a \in Field s. f a \neq g a}
<proof>

lemma *max-fun-diff-max*: $\llbracket f \neq g; f \in \text{FINFUNC}; g \in \text{FINFUNC}; x \in \{a \in \text{Field } s. f a \neq g a\} \rrbracket \implies$
(x, s.max-fun-diff f g) \in s
<proof>

lemma *max-fun-diff*:
 $\llbracket f \neq g; f \in \text{FINFUNC}; g \in \text{FINFUNC} \rrbracket \implies$
 $(\exists a b. a \neq b \wedge a \in \text{Field } r \wedge b \in \text{Field } r \wedge$
 $f (s.max-fun-diff f g) = a \wedge g (s.max-fun-diff f g) = b)$
<proof>

lemma *max-fun-diff-le-eq*:
 $\llbracket (s.max-fun-diff f g, x) \in s; f \neq g; f \in \text{FINFUNC}; g \in \text{FINFUNC}; x \neq s.max-fun-diff f g \rrbracket \implies$
 $f x = g x$
<proof>

lemma *max-fun-diff-max2*:
assumes *ineq*: *s.max-fun-diff f g = s.max-fun-diff g h \longrightarrow*

$f (s.max\text{-fun-diff } f g) \neq h (s.max\text{-fun-diff } g h)$ **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 $s.max\text{-fun-diff } f h = s.max2 (s.max\text{-fun-diff } f g) (s.max\text{-fun-diff } g h)$
(is $?fh = s.max2 ?fg ?gh)$
 <proof>

definition *oexp* **where**

$oexp = \{(f, g) . f \in FINFUNC \wedge g \in FINFUNC \wedge$
 $((let m = s.max\text{-fun-diff } f g \text{ in } (f m, g m) \in r) \vee f = g)\}$

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*: $Partial\text{-order}\ oexp$
 <proof>

lemma *oexp-Total*: $Total\ oexp$
 <proof>

lemma *oexp-Linear-order*: $Linear\text{-order}\ oexp$
 <proof>

definition *const* = $(\lambda x. \text{if } x \in Field\ s \text{ then } r.zero \text{ else } undefined)$

lemma *const-in[simp]*: $x \in Field\ s \implies const\ x = r.zero$
 <proof>

lemma *const-notin[simp]*: $x \notin Field\ s \implies const\ x = undefined$
 <proof>

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

lemma *const-FINFUNC[simp]*: $Field\ r \neq \{\} \implies const \in FINFUNC$
 <proof>

lemma *const-least*:

assumes $\text{Field } r \neq \{\}$ $f \in \text{FINFUNC}$

shows $(\text{const}, f) \in \text{oexp}$

<proof>

lemma *support-not-const*:

assumes $F \subseteq \text{FINFUNC}$ **and** $\text{const} \notin F$

shows $\forall f \in F. \text{finite } (\text{SUPP } f) \wedge \text{SUPP } f \neq \{\} \wedge \text{SUPP } f \subseteq \text{Field } s$

<proof>

lemma *maxim-isMaxim-support*:

assumes $F \subseteq \text{FINFUNC}$ **and** $\text{const} \notin F$

shows $\forall f \in F. s.\text{isMaxim } (\text{SUPP } f) (s.\text{maxim } (\text{SUPP } f))$

<proof>

lemma *oexp-empty2*: $\text{Field } s = \{\} \implies \text{oexp} = \{(\lambda x. \text{undefined}, \lambda x. \text{undefined})\}$

<proof>

lemma *oexp-empty*: $\llbracket \text{Field } r = \{\}; \text{Field } s \neq \{\} \rrbracket \implies \text{oexp} = \{\}$

<proof>

lemma *fun-upd-FINFUNC*: $\llbracket f \in \text{FINFUNC}; x \in \text{Field } s; y \in \text{Field } r \rrbracket \implies f(x := y) \in \text{FINFUNC}$

<proof>

lemma *fun-upd-same-oexp*:

assumes $(f, g) \in \text{oexp}$ $f x = g x$ $x \in \text{Field } s$ $y \in \text{Field } r$

shows $(f(x := y), g(x := y)) \in \text{oexp}$

<proof>

lemma *fun-upd-smaller-oexp*:

assumes $f \in \text{FINFUNC}$ $x \in \text{Field } s$ $y \in \text{Field } r$ $(y, f x) \in r$

shows $(f(x := y), f) \in \text{oexp}$

<proof>

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

<proof>

lemma *oexp-Well-order*: *Well-order* *oexp*

<proof>

interpretation *o*: *wo-rel* *oexp* *<proof>*

lemma *zero-oexp*: $\text{Field } r \neq \{\} \implies o.\text{zero} = \text{const}$

<proof>

end

notation *wo-rel2.oexp* (**infixl** $\hat{\sim} o$ 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*
 \langle *proof* \rangle

lemma *ozero-ordIso*[*simp*]: *ozero* =_o *ozero*
 \langle *proof* \rangle

lemma *Field-ozero*[*simp*]: *Field* *ozero* = $\{\}$
 \langle *proof* \rangle

lemma *iso-ozero-empty*[*simp*]: *r* =_o *ozero* = (*r* = $\{\}$)
 \langle *proof* \rangle

lemma *ozero-ordLeq*:
assumes *Well-order* *r* **shows** *ozero* ≤_o *r*
 \langle *proof* \rangle

definition *oone* = $\{((\cdot),(\cdot))\}$

lemma *oone-Well-order*[*simp*]: *Well-order* *oone*
 \langle *proof* \rangle

lemma *Field-oone*[*simp*]: *Field* *oone* = $\{(\cdot)\}$
 \langle *proof* \rangle

lemma *oone-ordIso*: *oone* =_o $\{(x,x)\}$
 \langle *proof* \rangle

lemma *osum-ordLeqR*: *Well-order* *r* \implies *Well-order* *s* \implies *s* ≤_o *r* +_o *s*
 \langle *proof* \rangle

lemma *osum-congL*:
assumes *r* =_o *s* **and** *t*: *Well-order* *t*
shows *r* +_o *t* =_o *s* +_o *t* (**is** ?*L* =_o ?*R*)
 \langle *proof* \rangle

lemma *osum-congR*:
assumes *r* =_o *s* **and** *t*: *Well-order* *t*
shows *t* +_o *r* =_o *t* +_o *s* (**is** ?*L* =_o ?*R*)
 \langle *proof* \rangle

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 $r \hat{=} \{\} = \{(\lambda x. \text{undefined}, \lambda x. \text{undefined})\}$
<proof>

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

lemma *oprod-zero[simp]: {} *o r = {} r *o {} = {}*
<proof>

lemma *oprod-congL:*
assumes $r =_o s$ **and** $t: \text{Well-order } t$
shows $r *o t =_o s *o t$ (**is** $?L =_o ?R$)
<proof>

lemma *oprod-congR:*
assumes $r =_o s$ **and** $t: \text{Well-order } t$
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 = { $\lambda x. \text{if } x \in \text{Field } s \text{ then } z \text{ else undefined}$ }*
<proof>

lemma *oone-ordIso-oexp:*
assumes $r =_o \text{oone}$ **and** $s: \text{Well-order } s$
shows $r \hat{=} s =_o \text{oone}$ (**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*: $\neg (s = o \text{ozero}) \implies \text{ozero} \hat{o} s = o \text{ozero}$

<proof>

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

<proof>

lemma *oexp-monoR*:

assumes $\text{oone} < o r$ $s < o t$

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 $\text{oone} < o r$

shows $s \leq o r \hat{o} s$

<proof>

lemma *FinFunc-osum*:

$fg \in \text{FinFunc } r (s + o t) = (fg \circ \text{Inl} \in \text{FinFunc } r s \wedge fg \circ \text{Inr} \in \text{FinFunc } r t)$

(**is** ?L = (?R1 \wedge ?R2))

<proof>

lemma *max-fun-diff-eq-Inl*:

assumes $\text{wo-rel.max-fun-diff } (s + o t) (\text{case-sum } f1 \ g1) (\text{case-sum } f2 \ g2) = \text{Inl } x$

$\text{case-sum } f1 \ g1 \neq \text{case-sum } f2 \ g2$

$\text{case-sum } f1 \ g1 \in \text{FinFunc } r (s + o t)$ $\text{case-sum } f2 \ g2 \in \text{FinFunc } r (s + o t)$

shows $\text{wo-rel.max-fun-diff } s \ f1 \ f2 = x$ (**is** ?P) $g1 = g2$ (**is** ?Q)

<proof>

lemma *max-fun-diff-eq-Inr*:

assumes $\text{wo-rel.max-fun-diff } (s + o t) (\text{case-sum } f1 \ g1) (\text{case-sum } f2 \ g2) = \text{Inr } x$

$\text{case-sum } f1 \ g1 \neq \text{case-sum } f2 \ g2$

$\text{case-sum } f1 \ g1 \in \text{FinFunc } r (s + o t)$ $\text{case-sum } f2 \ g2 \in \text{FinFunc } r (s + o t)$

shows $\text{wo-rel.max-fun-diff } t \ g1 \ g2 = x$ (**is** ?P) $g1 \neq g2$ (**is** ?Q)

<proof>

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

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

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

lemma *rev-curr-app-FinFunc[elim!]*:
 $\llbracket f \in \text{FinFunc } r (s * o t); z \in \text{Field } t \rrbracket \implies \text{rev-curr } f z \in \text{FinFunc } r s$
 ⟨proof⟩

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

lemma *oexp-oexp*: $(r \hat{o} s) \hat{o} t = o r \hat{o} (s * o t)$ (**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*: $\bigwedge x y y'. [(x,y) \in R; (x,y') \in R] \implies y = y'$
shows $|\{y. \exists x. (x,y) \in R\}| \leq_o |\{x. \exists y. (x,y) \in R\}|$
<proof>

lemma *card-of-unique2*: $[[\text{card-order-on } B \ r; \text{bij-betw } f \ A \ B]] \implies r =_o |A|$

<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** $\forall r \in R. \text{Card-order } r$

shows *Card-order (omax R)*

<proof>

lemma *Card-order-omax2*:

assumes *finite I and* $I \neq \{\}$

shows *Card-order (omax $\{A\ i \mid i. i \in I\}$)*

<proof>

10.2 Cardinals versus set operations on arbitrary sets

lemma *card-of-set-type[simp]*: $|UNIV::'a \text{ set}| <_o |UNIV::'a \text{ set set}|$

<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*:

$| \{\} :: 'b \text{ set} | <_o |A::'a \text{ set}| = (A \neq \{\})$

<proof>

lemma *card-of-empty5*:

$|A| <_o |B| \implies B \neq \{\}$

<proof>

lemma *Well-order-card-of-empty*:

Well-order $r \implies | \{\} | \leq_o r$

<proof>

lemma *card-of-UNIV[simp]*:

$|A :: 'a \text{ set}| \leq_o |UNIV :: 'a \text{ set}|$

$\langle \text{proof} \rangle$

lemma *card-of-UNIV2[simp]*:

$\text{Card-order } r \implies (r :: 'a \text{ rel}) \leq_o |UNIV :: 'a \text{ set}|$

$\langle \text{proof} \rangle$

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

assumes $|A| \leq_o |B|$

shows $|Pow A| \leq_o |Pow B|$

$\langle \text{proof} \rangle$

lemma *ordIso-Pow-mono[simp]*:

assumes $r \leq_o r'$

shows $|Pow(\text{Field } r)| \leq_o |Pow(\text{Field } r')|$

$\langle \text{proof} \rangle$

lemma *card-of-Pow-cong[simp]*:

assumes $|A| =_o |B|$

shows $|Pow A| =_o |Pow B|$

$\langle \text{proof} \rangle$

lemma *ordIso-Pow-cong[simp]*:

assumes $r =_o r'$

shows $|Pow(\text{Field } r)| =_o |Pow(\text{Field } r')|$

$\langle \text{proof} \rangle$

corollary *Card-order-Plus-empty1*:

$\text{Card-order } r \implies r =_o |(Field r) <+> \{\}|$

$\langle \text{proof} \rangle$

corollary *Card-order-Plus-empty2*:

$\text{Card-order } r \implies r =_o |\{\} <+> (Field r)|$

$\langle \text{proof} \rangle$

lemma *card-of-Un2[simp]*:

shows $|A| \leq_o |B \cup A|$

$\langle \text{proof} \rangle$

lemma *Un-Plus-bij-betw*:

assumes $A \text{ Int } B = \{\}$

shows $\exists f. \text{bij-betw } f (A \cup B) (A <+> B)$

$\langle \text{proof} \rangle$

lemma *card-of-Un-Plus-ordIso*:

assumes $A \text{ Int } B = \{\}$

shows $|A \cup B| =_o |A <+> B|$

$\langle \text{proof} \rangle$

lemma *card-of-Un-Plus-ordIso1*:

$$|A \cup B| =_o |A \langle + \rangle (B - A)|$$

<proof>

lemma *card-of-Un-Plus-ordIso2*:

$$|A \cup B| =_o |(A - B) \langle + \rangle B|$$

<proof>

lemma *card-of-Times-singl1*: $|A| =_o |A \times \{b\}|$

<proof>

corollary *Card-order-Times-singl1*:

$$\text{Card-order } r \implies r =_o |(Field\ r) \times \{b\}|$$

<proof>

lemma *card-of-Times-singl2*: $|A| =_o |\{b\} \times A|$

<proof>

corollary *Card-order-Times-singl2*:

$$\text{Card-order } r \implies r =_o |\{a\} \times (Field\ r)|$$

<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 $|A \times (\text{Field } r)| =_o |A \times (\text{Field } r')|$

<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 $|(\text{Field } r) \times (\text{Field } p)| =_o |(\text{Field } r') \times (\text{Field } p')|$

<proof>

lemma *card-of-Sigma-mono2*:

assumes *inj-on* f ($I::'i$ set) **and** $f' I \leq (J::'j$ set)

shows $|\text{SIGMA } i : I. (A::'j \Rightarrow 'a \text{ set}) (f i)| \leq_o |\text{SIGMA } j : J. A j|$

<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 $|\text{SIGMA } i : I. A (f i)| \leq_o |\text{SIGMA } j : J. B j|$

<proof>

lemma *ordLeq-Sigma-mono1*:

assumes $\forall i \in I. p i \leq_o r i$

shows $|\text{SIGMA } i : I. \text{Field}(p i)| \leq_o |\text{SIGMA } i : I. \text{Field}(r i)|$

<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 $|\text{SIGMA } i : I. \text{Field}(p(f i))| \leq_o |\text{SIGMA } j : J. \text{Field}(r j)|$

<proof>

lemma *ordIso-Sigma-cong1*:

assumes $\forall i \in I. p i =_o r i$

shows $|\text{SIGMA } i : I. \text{Field}(p i)| =_o |\text{SIGMA } i : I. \text{Field}(r i)|$

<proof>

lemma *ordLeq-Sigma-cong*:

assumes *bij-betw* $f I J$ **and**

$\forall j \in J. p j =_o r j$

shows $|\text{SIGMA } i : I. \text{Field}(p(f i))| =_o |\text{SIGMA } j : J. \text{Field}(r j)|$

<proof>

lemma *card-of-UNION-Sigma2*:

assumes $\bigwedge i j. [\{i,j\} \leq I; i \neq j] \implies A \ i \ \text{Int} \ A \ j = \{\}$
shows $|\bigcup_{i \in I}. A \ i| = o \ |\text{Sigma } I \ A|$

<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 $\exists f. \text{inj-on } f \ (A \ <+\> \ B) \wedge f \ ' \ (A \ <+\> \ B) \leq A \times B$

<proof>

corollary *Plus-into-Times-types*:

assumes $A2: (a1::'a) \neq a2$ **and** $B2: (b1::'b) \neq b2$
shows $\exists (f::'a \ + \ 'b \ \Rightarrow \ 'a \ * \ 'b). \ \text{inj } f$

<proof>

corollary *Times-same-infinite-bij-betw*:

assumes $\neg \text{finite } A$
shows $\exists f. \text{bij-betw } f \ (A \times A) \ A$

<proof>

corollary *Times-same-infinite-bij-betw-types*:

assumes $INF: \neg \text{finite}(UNIV::'a \ \text{set})$
shows $\exists (f::('a \ * \ 'a) \ \Rightarrow \ 'a). \ \text{bij } f$

<proof>

corollary *Times-infinite-bij-betw*:

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

<proof>

corollary *Times-infinite-bij-betw-types*:

assumes $\neg \text{finite}(UNIV::'a \ \text{set})$ **and** $\text{inj}(g::'b \ \Rightarrow \ 'a)$
shows $(\exists (f::('b \ * \ 'a) \ \Rightarrow \ 'a). \ \text{bij } f) \wedge (\exists (h::('a \ * \ 'b) \ \Rightarrow \ 'a). \ \text{bij } h)$

<proof>

lemma *card-of-Times-ordLeq-infinite*:

$[\neg \text{finite } C; |A| \leq o \ |C|; |B| \leq o \ |C|] \implies |A \times B| \leq o \ |C|$

<proof>

corollary *Plus-infinite-bij-betw*:

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

<proof>

corollary *Plus-infinite-bij-betw-types*:

assumes $\neg \text{finite}(UNIV::'a \ \text{set})$ **and** $\text{inj}(g::'b \ \Rightarrow \ 'a)$
shows $(\exists (f::('b \ + \ 'a) \ \Rightarrow \ 'a). \ \text{bij } f) \wedge (\exists (h::('a \ + \ 'b) \ \Rightarrow \ 'a). \ \text{bij } h)$

<proof>

lemma *card-of-Un-infinite*:

assumes *INF*: $\neg \text{finite } A$ **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]*:

$\llbracket \neg \text{finite } A; |B| \leq_o |A| \rrbracket \implies |A \cup B| =_o |A|$
 $\llbracket \neg \text{finite } A; |B| \leq_o |A| \rrbracket \implies |B \cup A| =_o |A|$
<proof>

lemma *card-of-Un-diff-infinite*:

assumes *INF*: $\neg \text{finite } A$ **and** *LESS*: $|B| <_o |A|$
shows $|A - B| =_o |A|$
<proof>

corollary *Card-order-Un-infinite*:

assumes *INF*: $\neg \text{finite}(\text{Field } r)$ **and** *CARD*: *Card-order* r **and**
LEQ: $p \leq_o r$
shows $|(\text{Field } r) \cup (\text{Field } p)| =_o r \wedge |(\text{Field } p) \cup (\text{Field } r)| =_o r$
<proof>

corollary *subset-ordLeq-diff-infinite*:

assumes *INF*: $\neg \text{finite } B$ **and** *SUB*: $A \leq B$ **and** *LESS*: $|A| <_o |B|$
shows $\neg \text{finite } (B - A)$
<proof>

lemma *card-of-Times-ordLess-infinite[simp]*:

assumes *INF*: $\neg \text{finite } C$ **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*: $\neg \text{finite}(\text{Field } r)$ **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* $(\text{Field } s)$
shows *finite* $(\text{Field } r)$
<proof>

lemma *ordIso-finite-Field*:

assumes $r =_o s$
shows *finite* $(\text{Field } r) \longleftrightarrow \text{finite}(\text{Field } s)$
<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 $\neg \text{finite } A$
shows $|\text{insert } a \ A| = o |A|$
<proof>

lemma *card-of-Un-singl-ordLess-infinite1*:

assumes $\neg \text{finite } B$ **and** $|A| < o |B|$
shows $|\{a\} \ \text{Un } A| < o |B|$
<proof>

lemma *card-of-Un-singl-ordLess-infinite*:

assumes $\neg \text{finite } B$
shows $|A| < o |B| \longleftrightarrow |\{a\} \ \text{Un } A| < o |B|$
<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 $nlists \ A \ n \equiv \{l. \ \text{set } l \leq A \wedge \text{length } l = n\}$

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 $\neg \text{finite } A$

shows $|\text{nlists } A \ n| \leq_o |A|$

$\langle \text{proof} \rangle$

lemma *Card-order-lists*: $\text{Card-order } r \implies r \leq_o |\text{lists}(\text{Field } r)|$

$\langle \text{proof} \rangle$

lemma *Union-set-lists*: $\bigcup (\text{set } ` (\text{lists } A)) = A$

$\langle \text{proof} \rangle$

lemma *inj-on-map-lists*:

assumes $\text{inj-on } f \ A$

shows $\text{inj-on } (\text{map } f) \ (\text{lists } A)$

$\langle \text{proof} \rangle$

lemma *map-lists-mono*:

assumes $f ` A \leq B$

shows $(\text{map } f) ` (\text{lists } A) \leq \text{lists } B$

$\langle \text{proof} \rangle$

lemma *map-lists-surjective*:

assumes $f ` A = B$

shows $(\text{map } f) ` (\text{lists } A) = \text{lists } B$

$\langle \text{proof} \rangle$

lemma *bij-betw-map-lists*:

assumes $\text{bij-betw } f \ A \ B$

shows $\text{bij-betw } (\text{map } f) \ (\text{lists } A) \ (\text{lists } B)$

$\langle \text{proof} \rangle$

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

assumes $|A| \leq_o |B|$

shows $|\text{lists } A| \leq_o |\text{lists } B|$

$\langle \text{proof} \rangle$

lemma *ordIso-lists-mono*:

assumes $r \leq_o r'$

shows $|\text{lists}(\text{Field } r)| \leq_o |\text{lists}(\text{Field } r')|$

$\langle \text{proof} \rangle$

lemma *card-of-lists-cong[simp]*:

assumes $|A| =_o |B|$

shows $|\text{lists } A| =_o |\text{lists } B|$

$\langle \text{proof} \rangle$

lemma *card-of-lists-infinite[simp]*:

assumes $\neg \text{finite } A$
shows $|\text{lists } A| =_o |A|$
 $\langle \text{proof} \rangle$

lemma *Card-order-lists-infinite*:
assumes *Card-order* r **and** $\neg \text{finite}(\text{Field } r)$
shows $|\text{lists}(\text{Field } r)| =_o r$
 $\langle \text{proof} \rangle$

lemma *ordIso-lists-cong*:
assumes $r =_o r'$
shows $|\text{lists}(\text{Field } r)| =_o |\text{lists}(\text{Field } r')|$
 $\langle \text{proof} \rangle$

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

corollary *lists-infinite-bij-betw-types*:
assumes $\neg \text{finite}(\text{UNIV} :: 'a \text{ set})$
shows $\exists (f :: 'a \text{ list} \Rightarrow 'a). \text{bij } f$
 $\langle \text{proof} \rangle$

10.5 Cardinals versus the finite powerset operator

lemma *card-of-Fpow[simp]*: $|A| \leq_o |\text{Fpow } A|$
 $\langle \text{proof} \rangle$

lemma *Card-order-Fpow*: *Card-order* $r \implies r \leq_o |\text{Fpow}(\text{Field } r)|$
 $\langle \text{proof} \rangle$

lemma *image-Fpow-surjective*:
assumes $f : A = B$
shows $(\text{image } f) : (\text{Fpow } A) = \text{Fpow } B$
 $\langle \text{proof} \rangle$

lemma *bij-betw-image-Fpow*:
assumes *bij-betw* $f A B$
shows *bij-betw* $(\text{image } f) (\text{Fpow } A) (\text{Fpow } B)$
 $\langle \text{proof} \rangle$

lemma *card-of-Fpow-mono[simp]*:
assumes $|A| \leq_o |B|$
shows $|\text{Fpow } A| \leq_o |\text{Fpow } B|$
 $\langle \text{proof} \rangle$

lemma *ordIso-Fpow-mono*:
assumes $r \leq_o r'$

shows $|Fpow(Field\ r)| \leq_o |Fpow(Field\ r^\wedge)|$
<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 $|Fpow(Field\ r)| =_o |Fpow(Field\ r^\wedge)|$
<proof>

lemma *card-of-Fpow-lists*: $|Fpow\ A| \leq_o |lists\ A|$
<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 $\exists f. bij_betw\ f\ (Fpow\ A)\ A$
<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-eg*:
 $n \leq m \implies wo-rel.ofilter\ (natLeq-on\ m)\ \{0 ..< n\}$
<proof>

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

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

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

lemma *natLeq-on-ordLess-natLeq*: $natLeq-on\ n <_o\ natLeq$
<proof>

lemma *natLeq-on-injective*:
 $natLeq-on\ m = natLeq-on\ n \implies m = n$
<proof>

lemma *natLeq-on-injective-ordIso*:
 $(natLeq-on\ m =_o\ natLeq-on\ n) = (m = n)$
<proof>

10.6.2 Then as cardinals

lemma *ordIso-natLeq-infinite1*:
 $|A| =_o\ natLeq \implies \neg finite\ A$
<proof>

lemma *ordIso-natLeq-infinite2*:
 $natLeq =_o\ |A| \implies \neg finite\ A$
<proof>

lemma *ordIso-natLeq-on-imp-finite*:
 $|A| =_o\ natLeq-on\ n \implies finite\ A$

<proof>

lemma *natLeq-on-Card-order: Card-order (natLeq-on n)*
<proof>

corollary *card-of-Field-natLeq-on:*
 $|Field (natLeq-on n)| =_o natLeq-on n$
<proof>

corollary *card-of-less:*
 $|\{0 ..< n\}| =_o natLeq-on n$
<proof>

lemma *natLeq-on-ordLeq-less-eq:*
 $((natLeq-on m) \leq_o (natLeq-on n)) = (m \leq n)$
<proof>

lemma *natLeq-on-ordLeq-less:*
 $((natLeq-on m) <_o (natLeq-on n)) = (m < n)$
<proof>

lemma *ordLeq-natLeq-on-imp-finite:*
assumes $|A| \leq_o natLeq-on n$
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 $(|A| \leq_o |B|) = (card A \leq card B)$
<proof>

lemma *finite-imp-card-of-natLeq-on:*
assumes *finite A*
shows $|A| =_o natLeq-on (card A)$
<proof>

lemma *finite-iff-card-of-natLeq-on:*
 $finite A = (\exists n. |A| =_o natLeq-on n)$
<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 $(|A| <_o |B|) = (\text{card } A < \text{card } B)$
 $\langle \text{proof} \rangle$

lemma *card-Field-natLeq-on*:
 $\text{card}(\text{Field}(\text{natLeq-on } n)) = n$
 $\langle \text{proof} \rangle$

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'*
 $r \text{ f}$
shows $r' =_o \text{Restr } r (f \text{ ' } (\text{Field } r'))$
 $\langle \text{proof} \rangle$

lemma *cardSuc-mono-ordLess[simp]*:
assumes *CARD*: Card-order *r* **and** *CARD'*: Card-order *r'*
shows $(\text{cardSuc } r <_o \text{cardSuc } r') = (r <_o r')$
 $\langle \text{proof} \rangle$

lemma *cardSuc-natLeq-on-Suc*:
 $\text{cardSuc}(\text{natLeq-on } n) =_o \text{natLeq-on}(\text{Suc } n)$
 $\langle \text{proof} \rangle$

lemma *card-of-Plus-ordLeq-infinite[simp]*:
assumes $\neg \text{finite } C$ **and** $|A| \leq_o |C|$ **and** $|B| \leq_o |C|$
shows $|A <+> B| \leq_o |C|$
 $\langle \text{proof} \rangle$

lemma *card-of-Un-ordLeq-infinite[simp]*:
assumes $\neg \text{finite } C$ **and** $|A| \leq_o |C|$ **and** $|B| \leq_o |C|$
shows $|A \text{ Un } B| \leq_o |C|$
 $\langle \text{proof} \rangle$

10.8 Others

lemma *under-mono[simp]*:
assumes Well-order *r* **and** $(i, j) \in r$
shows $\text{under } r \ i \subseteq \text{under } r \ j$
 $\langle \text{proof} \rangle$

lemma *underS-under*:
assumes $i \in \text{Field } r$
shows $\text{underS } r \ i = \text{under } r \ i - \{i\}$
 $\langle \text{proof} \rangle$

lemma *relChain-under*:
assumes Well-order *r*
shows $\text{relChain } r (\lambda i. \text{under } r \ i)$

<proof>

lemma *card-of-infinite-diff-finite:*

assumes \neg *finite* *A* **and** *finite* *B*

shows $|A - B| =_o |A|$

<proof>

lemma *infinite-card-of-diff-singl:*

assumes \neg *finite* *A*

shows $|A - \{a\}| =_o |A|$

<proof>

lemma *card-of-vimage:*

assumes $B \subseteq \text{range } f$

shows $|B| \leq_o |f^{-1} B|$

<proof>

lemma *surj-card-of-vimage:*

assumes *surj* *f*

shows $|B| \leq_o |f^{-1} B|$

<proof>

definition *Bpow* **where**

$Bpow\ r\ A \equiv \{X . X \subseteq A \wedge |X| \leq_o r\}$

lemma *Bpow-empty[simp]:*

assumes *Card-order* *r*

shows $Bpow\ r\ \{\} = \{\{\}\}$

<proof>

lemma *singl-in-Bpow:*

assumes *rc: Card-order* *r*

and *r: Field* $r \neq \{\}$ **and** *a: a* $\in A$

shows $\{a\} \in Bpow\ r\ A$

<proof>

lemma *ordLeq-card-Bpow:*

assumes *rc: Card-order* *r* **and** *r: Field* $r \neq \{\}$

shows $|A| \leq_o |Bpow\ r\ A|$

<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 $A B \equiv$
 $\{f. (\forall a. f a \neq \text{None} \longleftrightarrow a \in A) \wedge (\forall a \in A. \text{case } f a \text{ of } \text{Some } b \Rightarrow b \in B \mid \text{None} \Rightarrow \text{True})\}$

lemma *card-of-Func-option-Func*:
 $|Func\text{-option } A B| =_o |Func A B|$
 $\langle \text{proof} \rangle$

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

lemma *Func-Pfunc*:
 $Func\text{-option } A B \subseteq Pfunc A B$
 $\langle \text{proof} \rangle$

lemma *Pfunc-Func-option*:
 $Pfunc A B = (\bigcup A' \in Pow A. Func\text{-option } A' B)$
 $\langle \text{proof} \rangle$

lemma *card-of-Func-mono*:
fixes $A1 A2 :: 'a \text{ set}$ **and** $B :: 'b \text{ set}$
assumes $A12: A1 \subseteq A2$ **and** $B: B \neq \{\}$
shows $|Func A1 B| \leq_o |Func A2 B|$
 $\langle \text{proof} \rangle$

lemma *card-of-Func-option-mono*:
fixes $A1 A2 :: 'a \text{ set}$ **and** $B :: 'b \text{ set}$
assumes $A12: A1 \subseteq A2$ **and** $B: B \neq \{\}$
shows $|Func\text{-option } A1 B| \leq_o |Func\text{-option } A2 B|$
 $\langle \text{proof} \rangle$

lemma *card-of-Pfunc-Pow-Func-option*:
assumes $B \neq \{\}$
shows $|Pfunc A B| \leq_o |Pow A \times Func\text{-option } A B|$
 $\langle \text{proof} \rangle$

lemma *Bpow-ordLeq-Func-Field*:
assumes $rc: \text{Card-order } r$ **and** $r: \text{Field } r \neq \{\}$ **and** $A: \neg \text{finite } A$
shows $|Bpow r A| \leq_o |Func (Field r) A|$
 $\langle \text{proof} \rangle$

lemma *empty-in-Func[simp]*:
 $B \neq \{\} \implies (\lambda x. \text{undefined}) \in Func \{\} B$
 $\langle \text{proof} \rangle$

lemma *Func-mono[simp]*:

assumes $B1 \subseteq B2$
shows $Func\ A\ B1 \subseteq Func\ A\ B2$
 $\langle proof \rangle$

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

lemma *card-of-Func-UNIV-UNIV*:
 $|Func\ (UNIV::'a\ set)\ (UNIV::'b\ set)| =_o\ |UNIV::('a \Rightarrow 'b)\ set|$
 $\langle proof \rangle$

lemma *ordLeq-Func*:
assumes $\{b1, b2\} \subseteq B$ $b1 \neq b2$
shows $|A| \leq_o\ |Func\ A\ B|$
 $\langle proof \rangle$

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

10.9 Infinite cardinals are limit ordinals

lemma *card-order-infinite-isLimOrd*:
assumes $c: Card\ order\ r$ **and** $i: \neg finite\ (Field\ r)$
shows $isLimOrd\ r$
 $\langle proof \rangle$

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

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

lemma *Field-init-seg-of[simp]*:
 $Field\ init\ seg\ of = UNIV$
 $\langle proof \rangle$

lemma *refl-init-seg-of[intro, simp]*: $refl\ init\ seg\ of$
 $\langle proof \rangle$

lemma *regularCard-all-ex*:
assumes $r: Card\ order\ r$ $regularCard\ r$
and $As: \bigwedge i\ j\ b. b \in B \implies (i, j) \in r \implies P\ i\ b \implies P\ j\ b$

and *Bsub*: $\forall b \in B. \exists i \in \text{Field } r. P i b$
and *cardB*: $|B| < o r$
shows $\exists i \in \text{Field } r. \forall b \in B. P i b$
 <proof>

lemma *relChain-stabilize*:

assumes *rc*: *relChain* *r* *As* **and** *AsB*: $(\bigcup i \in \text{Field } r. As i) \subseteq B$ **and** *Br*: $|B| < o r$
and *ir*: $\neg \text{finite } (\text{Field } r)$ **and** *cr*: *Card-order* *r*
shows $\exists i \in \text{Field } r. As (\text{succ } r i) = As i$
 <proof>

10.10 Regular vs. stable cardinals

lemma *stable-cardSuc*:

assumes *CARD*: *Card-order* *r* **and** *INF*: $\neg \text{finite } (\text{Field } r)$
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*: $\forall r \in R. \text{Card-order } (r::'a \text{ rel})$
shows $\exists (A :: (\text{nat} + 'a \text{ set}) \text{ set}).$
 $\neg \text{finite } A \wedge \text{stable } |A| \wedge (\forall r \in R. r < o |A|)$
 <proof>

corollary *infinite-regularCard-exists*:

assumes *CARD*: $\forall r \in R. \text{Card-order } (r::'a \text{ rel})$
shows $\exists (A :: (\text{nat} + 'a \text{ set}) \text{ set}).$
 $\neg \text{finite } A \wedge \text{regularCard } |A| \wedge (\forall r \in R. r < o |A|)$
 <proof>

end

11 Cardinal Arithmetic

theory *Cardinal-Arithmetic*

imports *Cardinal-Order-Relation*

begin

11.1 Binary sum

lemma *csum-Cnotzero2*:

$Cnotzero\ r2 \implies Cnotzero\ (r1 + c\ r2)$

$\langle proof \rangle$

lemma *single-cone*:

$|\{x\}| = o\ cone$

$\langle proof \rangle$

lemma *cone-Cnotzero*: $Cnotzero\ cone$

$\langle proof \rangle$

lemma *cone-ordLeq-ctwo*: $cone \leq o\ ctwo$

$\langle proof \rangle$

lemma *csum-czero1*: $Card\text{-}order\ r \implies r + c\ czero = o\ r$

$\langle proof \rangle$

lemma *csum-czero2*: $Card\text{-}order\ r \implies czero + c\ r = o\ r$

$\langle proof \rangle$

11.2 Product

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

$\langle proof \rangle$

lemma *card-of-Times-singleton*:

fixes $A :: 'a\ set$

shows $|A \times \{x\}| = o\ |A|$

$\langle proof \rangle$

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

$\langle proof \rangle$

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

$\langle proof \rangle$

lemma *cprod-cone*: $Card\text{-}order\ r \implies r *c\ cone = o\ r$

$\langle proof \rangle$

lemma *ordLeq-cprod1*: $\llbracket Card\text{-}order\ p1; Cnotzero\ p2 \rrbracket \implies p1 \leq o\ p1 *c\ p2$

$\langle proof \rangle$

11.3 Exponentiation

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

$\langle proof \rangle$

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 $r \implies Cinfinite (ctwo \hat{c} r)$
(proof)

lemma *cone-ordLeq-iff-Field*:
assumes $cone \leq_o r$
shows *Field* $r \neq \{\}$
(proof)

lemma *cone-ordLeq-cexp*: $cone \leq_o r1 \implies cone \leq_o r1 \hat{c} r2$
(proof)

lemma *Card-order-czero*: *Card-order* $czero$
(proof)

lemma *cexp-mono2''*:
assumes $2: p2 \leq_o r2$
and $n1: Cnotzero q$
and $n2: Card-order p2$
shows $q \hat{c} p2 \leq_o q \hat{c} r2$
(proof)

lemma *csum-cexp*: $\llbracket Cinfinite r1; Cinfinite r2; Card-order q; ctwo \leq_o q \rrbracket \implies$
 $q \hat{c} r1 +_c q \hat{c} r2 \leq_o q \hat{c} (r1 +_c r2)$
(proof)

lemma *csum-cexp'*: $\llbracket Cinfinite r; Card-order q; ctwo \leq_o q \rrbracket \implies q +_c r \leq_o q \hat{c} r$
(proof)

lemma *card-of-Sigma-ordLeq-Cinfinite*:
 $\llbracket Cinfinite r; |I| \leq_o r; \forall i \in I. |A i| \leq_o r \rrbracket \implies |SIGMA i : I. A i| \leq_o r$
(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*: $Cnotzero\ r \implies czero \hat{c}\ r =_o czero$
<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*: $card\text{-}order\ r \implies card\text{-}order\ (cpow\ r)$
<proof>

lemma *cpow-greater-eq*: $Card\text{-}order\ r \implies r \leq_o cpow\ r$
<proof>

lemma *Cinfinite-cpow*: $Cinfinite\ r \implies Cinfinite\ (cpow\ r)$
<proof>

lemma *Card-order-cpow*: $Card\text{-}order\ (cpow\ r)$
<proof>

lemma *cardSuc-ordLeq-cpow*: $Card\text{-}order\ r \implies cardSuc\ r \leq_o cpow\ r$
<proof>

lemma *cpow-cexp-ctwo*: $cpow\ r =_o ctwo \hat{c}\ r$

<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 $r \leq_o \text{cmax } r \ s \wedge s \leq_o \text{cmax } r \ s$
 ⟨proof⟩

lemmas $\text{ordLeq-cmax1} = \text{ordLeq-cmax}[\text{THEN } \text{conjunct1}]$ **and**
 $\text{ordLeq-cmax2} = \text{ordLeq-cmax}[\text{THEN } \text{conjunct2}]$

lemma *finite-cmax*:

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

shows $\text{finite } (\text{Field } (\text{cmax } r \ s)) \longleftrightarrow \text{finite } (\text{Field } r) \wedge \text{finite } (\text{Field } s)$

⟨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*

$\text{downset-on } A \ r = (\forall x \ y. (x, y) \in r \wedge y \in A \longrightarrow x \in A)$

lemma *downset-onI*:

$(\bigwedge x \ y. (x, y) \in r \Longrightarrow y \in A \Longrightarrow x \in A) \Longrightarrow \text{downset-on } A \ r$

⟨proof⟩

lemma *downset-onD*:

$\text{downset-on } A \ r \Longrightarrow (x, y) \in r \Longrightarrow y \in A \Longrightarrow x \in A$

⟨proof⟩

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

definition *extension-on where*

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

lemma *extension-onI*:

$(\bigwedge x \ y. \llbracket x \in A; y \in A; (x, y) \in s \rrbracket \Longrightarrow (x, y) \in r) \Longrightarrow \text{extension-on } A \ r \ s$

⟨proof⟩

lemma *extension-onD*:

$\text{extension-on } A \ r \ s \Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow (x, y) \in s \Longrightarrow (x, y) \in r$

⟨proof⟩

lemma *downset-on-Union*:
assumes $\bigwedge r. r \in R \implies \text{downset-on } (\text{Field } r) p$
shows $\text{downset-on } (\text{Field } (\bigcup R)) p$
 $\langle \text{proof} \rangle$

lemma *chain-subset-extension-on-Union*:
assumes $\text{chain}_{\subseteq} R$ **and** $\bigwedge r. r \in R \implies \text{extension-on } (\text{Field } r) r p$
shows $\text{extension-on } (\text{Field } (\bigcup R)) (\bigcup R) p$
 $\langle \text{proof} \rangle$

lemma *downset-on-empty [simp]*: $\text{downset-on } \{\} p$
 $\langle \text{proof} \rangle$

lemma *extension-on-empty [simp]*: $\text{extension-on } \{\} p q$
 $\langle \text{proof} \rangle$

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

theorem *well-order-extension*:
assumes $\text{wf } p$
shows $\exists w. p \subseteq w \wedge \text{Well-order } w$
 $\langle \text{proof} \rangle$

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

corollary *total-well-order-extension*:
assumes $\text{wf } p$
shows $\exists w. p \subseteq w \wedge \text{Well-order } w \wedge \text{Field } w = \text{UNIV}$
 $\langle \text{proof} \rangle$

corollary *well-order-on-extension*:
assumes $\text{wf } p$ **and** $\text{Field } p \subseteq A$
shows $\exists w. p \subseteq w \wedge \text{well-order-on } A w$
 $\langle \text{proof} \rangle$

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 ⇒ 'b) ⇒ 'a set['k] ⇒ 'b set['k] **is** image
 ⟨proof⟩

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

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

lift-definition binsert :: 'a ⇒ 'a set['k] ⇒ '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| ⇒
 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 (∀ t ∈ set-bset a. ∃ u ∈ set-bset b. R t u) ∧ (∀ u ∈ set-bset b. ∃ t ∈ set-bset
 a. R t u) ↔
 ((BNF-Def.Grp {a. set-bset a ⊆ {(a, b). R a b}} (map-bset fst))⁻¹⁻¹ OO
 BNF-Def.Grp {a. set-bset a ⊆ {(a, b). R a b}} (map-bset snd)) a b **(is** ?L ↔
 ?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
 ⟨proof⟩

lemma map-bset-bempty[simp]: map-bset f bempty = bempty
 ⟨proof⟩

lemma *map-bset-binsert[simp]*: $\text{map-bset } f (\text{binsert } x X) = \text{binsert } (f x) (\text{map-bset } f X)$
 ⟨proof⟩

lemma *map-bset-bsingleton*: $\text{map-bset } f (\text{bsingleton } x) = \text{bsingleton } (f x)$
 ⟨proof⟩

lemma *bempty-not-binsert*: $\text{bempty} \neq \text{binsert } x X \text{ binsert } x X \neq \text{bempty}$
 ⟨proof⟩

lemma *bempty-not-bsingleton[simp]*: $\text{bempty} \neq \text{bsingleton } x \text{ bsingleton } x \neq \text{bempty}$
 ⟨proof⟩

lemma *bsingleton-inj[simp]*: $\text{bsingleton } x = \text{bsingleton } y \longleftrightarrow x = y$
 ⟨proof⟩

lemma *rel-bsingleton[simp]*:
 $\text{rel-bset } R (\text{bsingleton } x1) (\text{bsingleton } x2) = R x1 x2$
 ⟨proof⟩

lemma *rel-bset-bsingleton[simp]*:
 $\text{rel-bset } R (\text{bsingleton } x1) = (\lambda X. X \neq \text{bempty} \wedge (\forall x2 \in \text{set-bset } X. R x1 x2))$
 $\text{rel-bset } R X (\text{bsingleton } x2) = (X \neq \text{bempty} \wedge (\forall x1 \in \text{set-bset } X. R x1 x2))$
 ⟨proof⟩

lemma *rel-bset-bempty[simp]*:
 $\text{rel-bset } R \text{ bempty } X = (X = \text{bempty})$
 $\text{rel-bset } R Y \text{ bempty} = (Y = \text{bempty})$
 ⟨proof⟩

definition *bset-of-option where*
 $\text{bset-of-option} = \text{case-option bempty bsingleton}$

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

lemma *rel-bset-False[simp]*: $\text{rel-bset } (\lambda x y. \text{False}) x y = (x = \text{bempty} \wedge y = \text{bempty})$
 ⟨proof⟩

lemma *rel-bset-of-option[simp]*:
 $\text{rel-bset } R (\text{bset-of-option } x1) (\text{bset-of-option } x2) = \text{rel-option } R x1 x2$
 ⟨proof⟩

lemma *rel-bgraph[simp]*:
 $\text{rel-bset } (\text{rel-prod } (=) R) (\text{bgraph } f1) (\text{bgraph } f2) = \text{rel-fun } (=) (\text{rel-option } R) f1 f2$
 ⟨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 \longleftrightarrow X = bempty*

<proof>

lemma *map-bset-eq-bsingleton-iff*[simp]:

map-bset f X = bsingleton x \longleftrightarrow (set-bset X \neq {}) \wedge ($\forall y \in$ set-bset X. $f y = x$)
<proof>

lift-definition *bCollect* :: *('a \Rightarrow bool) \Rightarrow 'a set['a set] is Collect*

<proof>

lift-definition *bmember* :: *'a \Rightarrow 'a set['k] \Rightarrow bool is (\in)* *<proof>*

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

<proof>

lemma *bset-eq-iff*: *A = B \longleftrightarrow ($\forall a$. *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 \Longrightarrow*

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 (λA . *Ball (set-bset A) P*)*
<proof>

end

end