The HOL wellorder Library

J. R. Harrison

University of Cambridge Computer Laboratory
New Museums Site
Pembroke Street
Cambridge, CB2 3QG
England.

30th May 1992

Contents

Chapter 1

The wellorder library

The wellorder library develops a small theory of orderings, particularly wellorderings. This theory is used to prove a theorem justifying definitions by recursion over a wellordering. One can use an arbitrary recursion equation:

```
?!f. !a. f(a) = h f a                                                    1
```

provided one can show that the value of `h f a` depends only on the values of `f` for values with lower measure than `a`, where the measure is some function into the range of a wellordering.

The library also contains some frequently used forms of the Axiom Of Choice, namely:

- The Well-Ordering Principle: every set can be well-ordered.
- Hausdorff's Maximal Principle: every poset has a maximal chain.
- Zorn's Lemma: a poset in which every chain has an upper bound contains a maximal element.
- Kuratowski's Lemma: every chain in a poset is contained in a maximal chain.

The names attached to the above theorems are probably the most conventional, but are no indication of priority. In fact the 'maximal' principles were independently discovered by various people. The Well-Ordering theorem was first stated by Cantor, who regarded it as self-evident, and first proved by Zermelo using the Axiom Of Choice. More details of the history of the Axiom of Choice can be found in [4].

The HOL logic includes a strong form of the Axiom as standard (namely `SELECT_AX` which specifies the behaviour of ther Hilbert ε-operator). From this, the above four theorems are deduced. The proofs are not original: they mainly follow the proofs in the set theory chapter of Dudley's book [3], with a few ideas (e.g. the simple characterization of wellordering) taken from [5]. The latter book, like [2], proves the results in a different way starting from a fixpoint theorem for CPOs due to Bourbaki [1]. However the proof of that theorem is rather ugly.

1.1 Loading the library

Assuming HOL is installed properly, it is only necessary to execute the following ML command:

```
#load_library `wellorder`;;                                              2
```

The theorems and definitions are then set up to autoload as described latter in this document.

1.2 Basic definitions and theorems in the wellorder library

The theorems in the library are all couched in terms of ordering relations, so the library begins with some preliminary definitions concerned with ordered sets. To make it easier for a user to massage the theorems as required for a particular application, all the definitions are prefixed with `wo_` so they are unlikely to clash with any others. The library provides functions `set_wo_map` and `unset_wo_map` to set and remove an interface map to hide this from the user. In the succeeding discussion, we ignore `wo_` prefixes.

1.2.1 Definitions about sets

We consider predicates as sets rather than define a new type. In the following discussion, we will generally use set notation: $x \in P$ rather than Px or $P(x)$. The only two purely set-theoretic definitions made are:

- $P \subseteq Q \equiv \forall x.\ x \in P \Longrightarrow x \in Q$
- $\bigcup P \equiv \lambda x.\ \exists p.\ p \in P \wedge x \in p$

1.2.2 Definitions about orders

The usual definition of an ordered set is a pair $(P, \leq)$ where $\leq$ is a subset of the Cartesian product $P \times P$. Continual manipulation of pairs makes the proofs rather more awkward, so we identify an ordered set with the $\leq$ relation itself. (In the following, we will for example use the terms 'wellorder' and 'woset' without discrimination.) If required, the underlying set P can be recovered by the fl (field) function, defined as:

$$fl(\leq)x = \exists y.\ x \leq y \vee y \leq x$$

Since orders (partial or otherwise) are always symmetric, this is equivalent to

$$fl(\leq)x = x \leq x$$

However it is useful to have the above definition available to prove something *is* an order.

Now a partially ordered set (poset) is defined to be a binary relation which is reflexive, transitive and antisymmetric:

$$\begin{aligned} poset(\leq) = \; & (\forall x \in fl(\leq).\ x \leq x) \wedge \\ & (\forall xyz.\ x \leq y \wedge y \leq z \Longrightarrow x \leq z) \wedge \\ & (\forall xy.\ x \leq y \wedge y \leq x \Longrightarrow (x = y)) \end{aligned}$$

A well-ordered set (woset) has the two additional properties of being total and well-founded:

$$\begin{aligned} woset(\leq) = \; & (\forall x \in fl(\leq).\ x \leq x) \wedge \\ & (\forall xyz.\ x \leq y \wedge y \leq z \Longrightarrow x \leq z) \wedge \\ & (\forall xy.\ x \leq y \wedge y \leq x \Longrightarrow (x = y)) \wedge \\ & (\forall x \in fl(\leq),\ y \in fl(\leq).\ x \leq y \vee y \leq x)) \wedge \\ & (\forall P \subseteq fl(\leq).\ (P \neq \emptyset) \Longrightarrow \exists y \in fl(\leq).\ Py \wedge \forall z \in P.\ y \leq z) \end{aligned}$$

We also define a chain in a poset as a set whose elements are all comparable under the ordering, i.e.

$$chain(\leq)C = \forall x \in C,\ y \in C.\ x \leq y \vee y \leq x$$

It would be more consistent to define a chain as a total order which is a subset of $\leq$, but since we have little interest in the chain as an ordered set, we use a subset of the field instead.

Next is the property of one relation being an initial segment of another. This is defined as

$$\leq'\ inseg \leq = \forall xy.\ x \leq' y = x \leq y \wedge y \in fl(\leq')$$

A special form of initial segment is the set of all elements less than some given element, thus:

$$linseg(\leq)a = \{(x, y) | x \leq y \wedge y < a\}$$

Finally, it is convenient to have a function `less` which converts a reflexive partial order into an irreflexive one:

$$less(\leq)(x, y) = x \leq y \wedge x \neq y$$

We define an *ordinal* to be a woset such that each element is chosen canonically using the choice operator applied to the complement of its predecessors (compare the von Neumann definition in set theory where each element is actually its set of predecessors).

$$ordinal(\leq) = woset(\leq) \wedge \forall x \in fl(\leq).\ x = \varepsilon y.\ y \not< x$$

1.3 A Useful lemma

Before embarking on the main proofs, we prove a few lemmas. One of these is quite interesting and useful. The following theorem reduces from 5 to 2 the number of properties we need to establish to show that a relation is a wellorder. It states that a relation is a wellorder iff it is antisymmetric and wellfounded (in the sense of every set having a *least*, not just minimal, element).

$$\begin{aligned} woset(\leq) = \ & (\forall xy.\ x \leq y \wedge y \leq x \implies (x = y)) \wedge \\ & (\forall P \subseteq fl(\leq).\ (P \neq \emptyset) \implies \exists y \in fl(\leq).\ Py \wedge \forall z \in P.\ y \leq z) \end{aligned}$$

Proof: Clearly it is sufficient to prove that a relation $\leq$ that is antisymmetric and wellfounded is also total, reflexive and transitive. The relation is total because for any x and y in its field, the set $\{x, y\}$ has a least member, so either $x \leq y$ or $y \leq x$. Reflexivity is a special case of totality, where x and y are the same. Finally, if $\leq$ is not transitive, then there exist x, y and z in the field such that

$$x \leq y \wedge y \leq z \wedge x \not\leq z$$

By totality, $z \leq x$, so we have

$$x \leq y \wedge y \leq z \wedge z \leq x$$

Clearly $\{x, y, z\} \subseteq fl(\leq)$, and so has a least element w. There are now three essentially identical cases to consider, according as w is x, y or z. It is easy to show that in each case we get $x = y = z$, so $x \leq z$, which is a contradiction.

1.4 The Recursion Theorem

The Recursion theorem is proved in three stages:

- Any two functions which satisfy the recursion equation for some downward-closed subset of the measure (i.e. for all arguments whose measures are less than some value) agree on the common ‘domain’. This is an easy proof by induction.
- For any value, one can indeed define such a function which ‘locally’ satisfies the recursion equation. Again this can be proved by induction.

- Now a global function exists. This can be defined as the 'union' of all the local functions, since by the first theorem, they all agree on any common domain.

1.5 Main theorems in the wellorder library

We list here the four versions of the Axiom Of Choice, and sketch their proof.

1.5.1 The Well-Ordering principle

Every set can be wellordered, i.e.

$$\forall P.\ \exists \leq .\ woset(\leq) \wedge (fl(\leq) = P)$$

Proof: In fact the HOL proof first shows that the universal set of every type can be wellordered. Then one can order a subset simply by the restriction of the wellorder to that subset. This makes the proofs in HOL simpler, so in the description that follows, P is actually a universal set of some type.

The proof exploits the canonicality of ordinals. The crucial result is that for any two ordinals, one is an initial segment of the other. We first prove two theorems which are closely related to two basic properties of 'real' ordinals:

- The union of any set of ordinals is an ordinal
- Each ordinal which does not correspond to the type universe has a 'successor' which arises by adding on the canonically chosen element.

We also show that the union of initial segments of an order is again an initial segment. All these proofs are quite easy by using the definitions directly.

Now suppose $\leq$ and $\leq'$ are two ordinals. Take the union of the set of all ordinals which are initial segments of $\leq$ and $\leq'$. By the above remarks, this is also an ordinal and hence the largest ordinal which is an initial segment of both. We claim that in fact it must be either the whole of $\leq$ or the whole of $\leq'$, because if not, we could form a successor. Because of the canonicality property, this successor is also an initial segment of both $\leq$ and $\leq'$, which yields a contradiction.

Now our wellordering is simply the union of all ordinals (of that type). Because of the above property, this gives a welldefined ordinal, and it is easy to see again that it must be the whole type or we could form its successor.

1.5.2 Hausdorff's Maximal Principle

Every poset has a maximal chain (under inclusion), i.e.

$$\exists C.\ chain(\leq)C \wedge \forall C'.\ chain(\leq)C' \wedge C \subseteq C' \Longrightarrow (C' = C)$$

Proof: The idea is to use transfinite recursion to produce a maximal chain. If $fl(\leq)$ is empty we are finished (the empty set serves). Otherwise, wellorder the whole of the type using the wellordering principle; call this wellordering $\leq_w$, and let $\perp$ be the $\leq_w$-least element of $fl(\leq)$. Now we can define a function over the whole type by recursion as follows:

$$f(x) = \begin{cases} x & \text{if } x \in fl(\leq) \text{ and } \forall y.\ y <_w x \Longrightarrow x \leq f(y) \vee f(y) \leq x \\ \perp & \text{otherwise} \end{cases}$$

We claim that the image under f of the whole type is a chain, i.e. that $f(x)$ and $f(y)$ are $\leq$-comparable for any x and y. Firstly it is easy to see that $f(\perp) = \perp$, and by construction every $f(x)$ is $\leq$-comparable with $\perp$. Furthermore each $f(x)$ is in $fl(\leq)$. Now it is quite easy to see that the result holds. If $x = y$ then it is immediate that $f(x) \leq f(y)$ as $\leq$ is reflexive on its field. Otherwise suppose $y <_w x$ (the other case is similar). By construction either $f(x) = \perp$, in which case we have already seen that comparability holds, or $f(x) = x$ and we know that x (and hence $f(x)$) is comparable to $f(y)$ since $y <_w x$. Consequently the range of f is a chain, and it is easy to see that it must be a maximal one, because if z is comparable to all members of the chain, it is certainly comparable to those elements $<_w$-less than it. Hence by construction it would have been added to the chain.

1.5.3 Zorn's Lemma

A poset in which each chain has an upper bound has a maximal element.

$$\begin{array}{l}(\forall C.\ chain(\leq)C \Longrightarrow \exists m \in fl(\leq).\ \forall x \in C.\ x \leq m) \\ \Longrightarrow \exists m \in fl(\leq).\ \forall x \in fl(\leq).\ m \leq x \Longrightarrow (m = x)\end{array}$$

Proof: Suppose $\leq$ is a poset with the property that every chain has an upper bound. By Hausdorff's Maximal Principle, we can find a maximal chain C, and by hypothesis, we can find an upper bound for it, say m.

We claim that m is a maximal element. For suppose there is an $x \in fl(\leq)$ with $m \leq x$. Evidently $C \cup \{x\}$ is then a chain, but since C is a *maximal* chain, we must have $C \cup \{x\} = C$, in other words $x \in C$. But since m is an upper bound for C, we have $x \leq m$. Now by antisymmetry of $\leq$, we have $m = x$ as required.

1.5.4 Kuratowski's Lemma

Every chain in a poset is contained in a maximal chain (under inclusion), i.e.

$$\begin{aligned}\forall C.\ chain(\leq)C \Longrightarrow \exists C'.\ \ & (chain(\leq)C' \wedge C \subseteq C') \wedge \\ & \forall C''.\ (chain(\leq)C'' \wedge C' \subseteq C'') \Longrightarrow (C'' = C')\end{aligned}$$

Proof: This could have been proved by a slight elaboration of the proof of the Hausdorff Maximal Principle, or by applying that principle to the partial order of elements not in, but comparable to all elements of, the chain. However the simplest seems to be a direct application of Zorn's lemma. Simply order the set of chains extending C. Zorn's Lemma assures us of a $\subseteq$-maximal chain extending C, provided we can show that a chain P of chains extending C has another such chain as its upper bound. But this is easy: if P is empty then take C, otherwise take the union of P.

Chapter 2

ML Functions in the wellorder Library

This chapter provides documentation on all the ML functions that are made available in HOL when the `wellorder` library is loaded. This documentation is also available online via the `help` facility.

set_wo_map

```
set_wo_map : (void -> void)
```

Synopsis

Sets interface map for constants in the “wellorder” library.

Description

To increase flexibility, the constants used in the wellorder library are declared with a `wo_` prefix. The `set_wo_map` function sets up an interface map to eliminate them from view. Any existing interface map is preserved alongside. The constants which have their prefixes hidden are: `subset`, `Union`, `fl`, `poset`, `chain`, `woset` and `inseg`.

Failure

Fails if an entry in the new interface map clashes with an existing one.

Comments

The interface map is not set up by default when the library is loaded.

See also

`interface_map`, `set_interface_map`, `unset_wo_map`.

unset_wo_map

```
unset_wo_map : (void -> void)
```

Synopsis
Removes interface map for constants in the "wellorder" library.

Description
To increase flexibility, the constants used in the wellorder library are declared with a `wo_` prefix. The `unset_wo_map` function cancels the effect of `set_wo_map`, which sets up an interface map to eliminate them from view. Any existing interface map is preserved when the wellorder-related entries are eliminated. The constants which have their prefixes hidden are: `subset`, `Union`, `fl`, `poset`, `chain`, `woset` and `inseg`.

Failure
Fails only if some entries in the interface map matching the above have been set by other means, so if subtracted, a constant would be hidden.

Comments
The interface map is not set up by default when the library is loaded.

See also
`interface_map`, `set_interface_map`, `set_wo_map`.

Chapter 3

Pre-proved Theorems

The sections that follow list all theorems in the `wellorder` library, including definitions. The theorems are grouped into definitions, lemmas, and the four main theorems. When the `wellorder` library is loaded, only the definitions and four main theorems are set up to autoload when their names are mentioned in ML. Lemmas, if needed, may be recovered by using the `theorem` function provided by ML. Note that all theorems are printed with the interface map set up by `set_wo_map` is in effect.

3.1 Basic definitions

```
wo_chain  (WELLORDER)
  |- !l P. chain l P = (!x y. P x /\ P y ==> l(x,y) \/ l(y,x))

wo_fl  (WELLORDER)
  |- !l x. fl l x = (?y. l(x,y) \/ l(y,x))

wo_inseg  (WELLORDER)
  |- !l m. l inseg m = (!x y. l(x,y) = m(x,y) /\ fl l y))

wo_linseg  (WELLORDER)
  |- !l a. linseg l a = (\(x,y). l(x,y) /\ less l(y,a)))

wo_ordinal  (WELLORDER)
  |- !l.
      ordinal l = woset l /\ (!x. fl l x ==> (x = (@y. ~less l(y,x)))))

wo_poset  (WELLORDER)
  |- !l.
      poset l =
      (!x. fl l x ==> l(x,x)) /\
      (!x y z. l(x,y) /\ l(y,z) ==> l(x,z)) /\
      (!x y. l(x,y) /\ l(y,x) ==> (x = y))

wo_subset  (WELLORDER)
  |- !P Q. P subset Q = (!x. P x ==> Q x)
```

```
wo_Union  (WELLORDER)
  |- !P. Union P = (\x. ?p. P p /\ p x)

wo_woset  (WELLORDER)
  |- !l.
       woset l =
       (!x. fl l x ==> l(x,x)) /\
       (!x y z. l(x,y) /\ l(y,z) ==> l(x,z)) /\
       (!x y. l(x,y) /\ l(y,x) ==> (x = y)) /\
       (!x y. fl l x /\ fl l y ==> l(x,y) \/ l(y,x)) /\
       (!P.
         (!x. P x ==> fl l x) /\ (?x. P x) ==>
         (?y. P y /\ (!z. P z ==> l(y,z))))
```

3.2 Miscellaneous lemmas

```
AGREE_LEMMA  (WELLORDER)
  |- !l h ms m n f g z.
       woset l /\
       (!x. fl l(ms x)) /\
       (!f f' x.
         (!y. less l(ms y,ms x) ==> (f y = f' y)) ==> (h f x = h f' x)) /\
       (!x. l(ms x,m) ==> (f x = h f x)) /\
       (!x. l(ms x,n) ==> (g x = h g x)) /\
       l(ms z,m) /\
       l(ms z,n) ==>
       (f z = g z)

CHAIN_SUBSET  (WELLORDER)
  |- !l P Q. chain l P /\ Q subset P ==> chain l Q

EXTEND_FL  (WELLORDER)
  |- !l x. woset l ==> (fl(\(x,y). l(x,y) /\ l(y,a))x = l(x,a))

EXTEND_INSEG  (WELLORDER)
  |- !l a. woset l /\ fl l a ==> (\(x,y). l(x,y) /\ l(y,a)) inseg l

EXTEND_LINSEG  (WELLORDER)
  |- !l a.
       woset l /\ fl l a ==>
       (\(x,y).
         linseg l a(x,y) \/ (y = a) /\ (fl(linseg l a)x \/ (x = a))) inseg
       l
```

```
FL_NUM (WELLORDER)
  |- !n. fl(\(m,n). m <= n)n

FL_SUC (WELLORDER)
  |- !l a.
      fl(\(x,y). l(x,y) \/ (y = a) /\ (fl l x \/ (x = a)))x =
      fl l x \/ (x = a)

INSEG_LINSEG (WELLORDER)
  |- !l m.
      woset l ==>
      (m inseg l = (m = l) \/ (?a. fl l a /\ (m = linseg l a)))

INSEG_PROPER_SUBSET (WELLORDER)
  |- !l m. m inseg l /\ ~(l = m) ==> (?x y. l(x,y) /\ ~m(x,y))

INSEG_PROPER_SUBSET_FL (WELLORDER)
  |- !l m. m inseg l /\ ~(l = m) ==> (?a. fl l a /\ ~fl m a)

INSEG_SUBSET (WELLORDER)
  |- !l m. m inseg l ==> (!x y. m(x,y) ==> l(x,y))

INSEG_SUBSET_FL (WELLORDER)
  |- !l m. m inseg l ==> (!x. fl m x ==> fl l x)

INSEG_WOSET (WELLORDER)
  |- !l m. m inseg l /\ woset l ==> woset m

KL_POSET_LEMMA (WELLORDER)
  |- poset(\(c1,c2). C subset c1 /\ c1 subset c2 /\ chain l c2)

LINSEG_FL (WELLORDER)
  |- !l a x. woset l ==> (fl(linseg l a)x = less l(x,a))

LINSEG_INSEG (WELLORDER)
  |- !l a. woset l ==> (linseg l a) inseg l

LINSEG_WOSET (WELLORDER)
  |- !l a. woset l ==> woset(linseg l a)

ORDINAL_CHAINED (WELLORDER)
  |- !l m. ordinal l /\ ordinal m ==> m inseg l \/ l inseg m
```

```
ORDINAL_SUC  (WELLORDER)
  |- !l.
       ordinal l /\ (?x. ~fl l x) ==>
       ordinal
       (\(x,y).
         l(x,y) \/ (y = (@y. ~fl l y)) /\ (fl l x \/ (x = (@y. ~fl l y))))

ORDINAL_UNION  (WELLORDER)
  |- !P. (!l. P l ==> ordinal l) ==> ordinal(Union P)

ORDINAL_UNION_LEMMA  (WELLORDER)
  |- !l x. ordinal l ==> fl l x ==> fl(Union ordinal)x

ORDINAL_UP  (WELLORDER)
  |- !l.
       ordinal l ==>
       (!x. fl l x) \/ (?m x. ordinal m /\ fl m x /\ ~fl l x)

PAIRED_EXT  (WELLORDER)
  |- !l m. (!x y. l(x,y) = m(x,y)) = (l = m)

POSET_FLEQ  (WELLORDER)
  |- !l. poset l ==> (!x. fl l x = l(x,x))

SUBSET_ANTISYM  (WELLORDER)
  |- !P Q. P subset Q /\ Q subset P ==> (P = Q)

SUBSET_REFL  (WELLORDER)
  |- !P. P subset P

SUBSET_TRANS  (WELLORDER)
  |- !P Q R. P subset Q /\ Q subset R ==> P subset R

UNION_FL  (WELLORDER)
  |- !P l. fl(Union P)x = (?l. P l /\ fl l x)

UNION_INSEG  (WELLORDER)
  |- !P l. (!m. P m ==> m inseg l) ==> (Union P) inseg l

WOSET  (WELLORDER)
  |- !l.
       woset l =
       (!x y. l(x,y) /\ l(y,x) ==> (x = y)) /\
       (!P.
         (!x. P x ==> fl l x) /\ (?x. P x) ==>
         (?y. P y /\ (!z. P z ==> l(y,z))))
```

```
WOSET_FLEQ  (WELLORDER)
  |- !l. woset l ==> (!x. fl l x = l(x,x))

WOSET_NUM  (WELLORDER)
  |- woset(\(m,n). m <= n)

WOSET_POSET  (WELLORDER)
  |- !l. woset l ==> poset l

WOSET_TOTAL_LE  (WELLORDER)
  |- !l. woset l ==> (!x y. fl l x /\ fl l y ==> l(x,y) \/ less l(y,x))

WOSET_TOTAL_LT  (WELLORDER)
  |- !l.
      woset l ==>
      (!x y. fl l x /\ fl l y ==> (x = y) \/ less l(x,y) \/ less l(y,x))

WOSET_TRANS_LE  (WELLORDER)
  |- !l. woset l ==> (!x y z. l(x,y) /\ less l(y,z) ==> less l(x,z))

WOSET_TRANS_LESS  (WELLORDER)
  |- !l. woset l ==> (!x y z. less l(x,y) /\ l(y,z) ==> less l(x,z))

WOSET_WELL_CONTRAPOS  (WELLORDER)
  |- !l.
      woset l ==>
      (!P.
        (!x. P x ==> fl l x) /\ (?x. P x) ==>
        (?y. P y /\ (!z. less l(z,y) ==> ~P z)))

WO_FL_RESTRICT  (WELLORDER)
  |- !l.
      woset l ==>
      (!P. fl(\(x,y). P x /\ P y /\ l(x,y))x = P x /\ fl l x)

WO_INDUCT  (WELLORDER)
  |- !P l.
      woset l /\ (!x. fl l x /\ (!y. less l(y,x) ==> P y) ==> P x) ==>
      (!x. fl l x ==> P x)

WO_LEMMA  (WELLORDER)
  |- ?l. ordinal l /\ (!x. fl l x)
```

```
WO_RECURSE  (WELLORDER)
  |- !l h ms.
       woset l /\
       (!x. fl l(ms x)) /\
       (!f g x.
         (!y. less l(ms y,ms x) ==> (f y = g y)) ==> (h f x = h g x)) ==>
       (?! f. !x. f x = h f x)

WO_RECURSE_EXISTS  (WELLORDER)
  |- !l h ms.
       woset l /\
       (!x. fl l(ms x)) /\
       (!f f' x.
         (!y. less l(ms y,ms x) ==> (f y = f' y)) ==> (h f x = h f' x)) ==>
       (?f. !x. f x = h f x)

WO_RECURSE_LOCAL  (WELLORDER)
  |- !l h ms.
       woset l /\
       (!x. fl l(ms x)) /\
       (!f f' x.
         (!y. less l(ms y,ms x) ==> (f y = f' y)) ==> (h f x = h f' x)) ==>
       (!n. ?f. !x. l(ms x,n) ==> (f x = h f x))

WO_RECURSE_NUM  (WELLORDER)
  |- !h ms.
       (!f g x.
         (!y. (ms y) < (ms x) ==> (f y = g y)) ==> (h f x = h g x)) ==>
       (?! f. !x. f x = h f x)
```

3.3 Main theorems

```
HP  (WELLORDER)
  |- !l.
       poset l ==>
       (?P. chain l P /\ (!Q. chain l Q /\ P subset Q ==> (Q = P)))

KL  (WELLORDER)
  |- !l.
       poset l ==>
       (!Q.
         chain l Q ==>
         (?P.
           (chain l P /\ Q subset P) /\
           (!R. chain l R /\ P subset R ==> (R = P))))
```

```
WO  (WELLORDER)
   |- !P. ?l. woset l /\ (fl l = P)

ZL  (WELLORDER)
   |- !l.
       poset l /\ (!P. chain l P ==> (?y. fl l y /\ (!x. P x ==> l(x,y)))) ==>
       (?y. fl l y /\ (!x. l(y,x) ==> (y = x)))
```

References

[1] N. Bourbaki, *Elements of mathematics, vol. 1: Set Theory*, Addison-Wesley 1974.

[2] B. A. Davey, H. A. Priestley, *Introduction to lattices and order*, Cambridge University Press 1990.

[3] R. M. Dudley, *Real analysis and probability*, Wadsworth and Brooks/Cole 1989.

[4] G. H. Moore, *Zermelo's axiom of choice: its origins, development, and influence*, Springer-Verlag 1982.

[5] R. R. Stoll, *Set theory and logic*, Dover 1979.

Index