Infinitary Induction in HOL

Roger Bishop Jones

Abstract

This paper explores some ideas for providing general support in HOL for structures defined by transfinite induction, by exploiting a strong infinity axiom expressed in terms of a well-ordering on a new type of "'a ordinals".

Created 2013/01/20

Last Change Date: 2014/08/17 16:07:53

http://www.rbjones.com/rbjpub/pp/doc/t051.pdf

Id: t051.doc,v 1.6 2014/08/17 16:07:53 rbj Exp

Contents

1 INTRODUCTION

My first ideas on providing support for defining structures by transfinite induction was to introduce an axiom of infinity in the constext of a new type of 'a ordinals and to use that to provide a type of infinitary trees. I did not get very far in that approach.

Later I returned and sought a fuller understanding of how to use such a strong axiom of infinity to introduce new large types, addressing the construction of V from Ord. This lead to a more general understanding of how one can use Ord directly in the construction of structures defined by transfinite induction.

On the basis of that understanding it became apparent to me that the introduction of polymorphic transfinite inductive types required a polymorphic type of 'a ordinals, in which the cardinality of the 'a ordinals depends on the cardinality of the type parameter (which might be thought of as representing a well-ordered collection of urelements).

The present work in progress on this document is the adding of this polymorphism, in anticipation of provision of some generic facility for introducing structures by transfinite induction.

The idea of developing a type of infinitary trees is now discarded in favour of working directly from the 'a ordinals.

2 PRELIMINARIES

This section is likely to mostly disappear. It will then contain those materials on 'a ordinals and cardinals which are independent of the types introduced in the next section.

SML

```
open_theory "rbjmisc";
force_new_theory "ordcard0";
new_parent "U_orders";
new_parent "trees";
new_parent "wf_relp";
new_parent "wf_recp";
force_new_pc "'ordcard0";
merge_pcs ["'savedthm_cs_∃_proof"] "'ordcard0";
set_merge_pcs ["rbjmisc", "'ordcard0"];
```

The material in this section is moved here en-block from t009 [1], and was not therefore originally undertaken for the purposes in hand. However, since I did not make use of it for any other purpose I now propose to use some of it here, expand the useful aspects, and discard some of the more obviously otiose material.

It is a treatment of cardinality as a property of sets which does not get so far as establishing types of 'a ordinals or cardinals. The definitions and theorems here and now considered as preliminaries to the establishment of 'a ordinal and cardinal numbers in a way not originally envisages, in the following sections.

The original motivation is in fact not far removed from the present motivation, which is nice ways of expressing strong axioms of infinity. Of course, the niceness which is most desirable is in the application of such axioms rather than in the aesthetics of their statement, and at the time when I starting the material in this section I didn't have much clue about the application.

The document as a whole reflects my present feeling that the applications (at least those of particular interest to me, but possible more generally) are best mediated by types of infinitary sequences and infinitary trees, and that other aspects of the set theories in which strong axioms are usually placed are less important in this context. In particular, whereas I had at times felt that the development of the treatment of functions was important, I now feel that it is not, and that the notion of function already available in HOL is sufficient. So the whole business of coding up functions as graphs of ordered pairs in set theory now seems unnecessary (*in this context*).

From here on in we have the original commentary (at least, *pro-tem*), which may not be entirely appropriate here.

The material in this section is primarily oriented towards expressing strong axioms of infinity. It does this by treating well-orderings as if they were 'a ordinals and cardinals.

The relations defined here with subscript s on their names are cardinality comparisons on sets.

SML

```
declare_infix(300, "≤s");
declare_infix(300, "<s");
declare_infix(300, "∼s");
```

HOL Constant

$$\$\leqslant_s : 'a\ SET \to 'b\ SET \to BOOL$$

$$\forall\ A\ B \bullet\ A \leqslant_s B \Leftrightarrow \exists f \bullet$$
$$\forall x\ y \bullet\ x \in A \wedge y \in A \Rightarrow f\ x \in B \wedge f\ y \in B \wedge (f\ x = f\ y \Rightarrow x = y)$$

$$\leqslant_s\_\mathbf{refl} =$$
$$\vdash \forall\ A \bullet\ A \leqslant_s A$$
$$\subseteq\_\leqslant_s\_\mathbf{thm} =$$
$$\vdash \forall\ A\ B \bullet\ A \subseteq B \Rightarrow A \leqslant_s B$$
$$\leqslant_s\_\mathbf{trans} =$$
$$\vdash \forall\ A\ B\ C \bullet\ A \leqslant_s B \wedge B \leqslant_s C \Rightarrow A \leqslant_s C$$

HOL Constant

$$\$<_s : 'a\ SET \to 'b\ SET \to BOOL$$

$$\forall\ A\ B \bullet\ A <_s B \Leftrightarrow A \leqslant_s B \wedge \neg\ B \leqslant_s A$$

$$\mathbf{lt}_s\_\mathbf{irrefl} =$$
$$\vdash \forall\ A \bullet \neg\ A <_s A$$
$$\mathbf{lt}_s\_\mathbf{trans} =$$
$$\vdash \forall\ A\ B\ C \bullet\ A <_s B \wedge B <_s C \Rightarrow A <_s C$$
$$\mathbf{lt}_s\_\leqslant_s\_\mathbf{trans} =$$
$$\vdash \forall\ A\ B\ C \bullet\ A <_s B \wedge B \leqslant_s C \Rightarrow A <_s C$$
$$\leqslant_s\_\mathbf{lt}_s\_\mathbf{trans} =$$
$$\vdash \forall\ A\ B\ C \bullet\ A \leqslant_s B \wedge B <_s C \Rightarrow A <_s C$$

HOL Constant

$$\$\sim_{\boldsymbol{s}} : 'a\ SET \to 'b\ SET \to BOOL$$

$$\forall\ A\ B\bullet$$
$$A \sim_s B \Leftrightarrow \exists f\ g\bullet$$
$$(\forall x\bullet\ x \in A \Rightarrow f\ x \in B \wedge g\ (f\ x) = x)$$
$$\wedge\quad (\forall y\bullet\ y \in B \Rightarrow g\ y \in A \wedge f\ (g\ y) = y)$$

$$\boldsymbol{card\_equiv\_lemma} =$$
$$\vdash \forall\ x\ y\ z\bullet\ x \sim_c x \wedge (x \sim_c y \Leftrightarrow y \sim_c x) \wedge (x \sim_c y \wedge y \sim_c z \Rightarrow x \sim_c z)$$

SML

```
commit_pc "'ordcard0";
force_new_pc "ordcard0";
merge_pcs ["rbjmisc", "'ordcard0"] "ordcard0";
commit_pc "ordcard0";
force_new_pc "ordcard01";
merge_pcs ["rbjmisc1", "'ordcard0"] "ordcard01";
commit_pc "ordcard01";
```

3 ORDINALS

I had at first intended to do a minimal set theory sufficient for defining a type of infinitary trees. But I think the simplest development is to go straight to a type of 'a ordinals and work forward from there.

SML

```
open_theory "ordcard0";
force_new_theory "ordcard";
new_parent "U_orders";
new_parent "wf_relp";
new_parent "wf_recp";
force_new_pc "'ordcard";
merge_pcs ["'savedthm_cs_∃_proof"] "'ordcard";
set_merge_pcs ["ordcard0", "'ordcard"];
```

The method is as follows. First introduce a type of 'a ordinals, then a type of cardinals which assists in formulation of a strong axiom of infinity.

3.1 The Type of Ordinals

SML

```
new_type ("ordinal", 1);
```

The purpose of the type parameter is to allow a strict lower bound to be placed on the cardinality of the type. This is necessary for support of polymorphic datatypes, since otherwise, however large the type of 'a ordinals the polymorphic datatype could be instantiated to a size larger than the 'a ordinals by supplying the 'a ordinals as a type parameter. With the polymorphic 'a ordinals we can always get a sufficiently large set of 'a ordinals by supplying to the type of 'a ordinals the same parameter give to the polymorphic datatype (or the product of multiple parameters).

The desired effect is as given by the following axiom:

SML

```
val card_slb = new_axiom(["card_slb"], ⌜
      (Universe:'a SET) <s (Universe:'a ordinal SET)
⌝);
```

We now use a well ordering theorem to define the ordering over the 'a ordinals. The consistency proof uses definitions and results from t009 [1]. The principal result is that every set can be well-ordered, but the definition of well-ordering does not entail well-foundedness or transitivity, since a well-ordering might be reflexive but well-foundedness does not admit reflexiveness. So the proof (not shown) takes an arbitrary well-ordering makes it irreflexive and then proves that the result is a well-founded well-ordering.

HOL Constant

```
<o: 'a ordinal → 'a ordinal  → BOOL
```

```
      WellOrdering(Universe, <o)
      ∧ WellFounded(Universe, <o)
```

SML

```
declare_infix(300, "<o");
```

It proves helpful to have this alternative rendering of well-foundedness:

```
lto_wf =
      ⊢ well_founded $<o
```

SML

```
val ORD_INDUCTION_T = WF_INDUCTION_T lto_wf;
val ord_induction_tac = wf_induction_tac lto_wf;
```

Every well-founded well-ordering is an initial segment of 'a ordinals, so we have now a type of 'a ordinals. At this point we have no idea how many 'a ordinals there are in the type, there might be only one.

So we will need a strong axiom of infinity to tell us that we have enough 'a ordinals for our purposes.

By analogy with a set theory with Universes, I assert that every 'a ordinal is less than some inaccessible 'a ordinal. To get an analogue to global replacement (rather than replacement below any inaccessible 'a ordinal corresponding to replacement within a "universe"), we would need to require that the universe is regular.

To help in expressing the notion of strong limit 'a ordinal the following definition is helpful:

The cardinality of a Von Neumann 'a ordinal is the cardinality of the collection of strictly smaller 'a ordinals. The following function which delivers that set. I also use the partial ordering of sets by cardinality ($<_s$) which was defined above.

SML

```
declare_infix(300, "≤o");
```

HOL Constant

$$\$\leqslant_{\boldsymbol{o}}: {}'a\ ordinal \to {}'a\ ordinal \to BOOL$$

$$\forall \beta\ \gamma \bullet\ \beta \leqslant_o \gamma \Leftrightarrow\ \beta <_o \gamma \vee \beta = \gamma$$

HOL Constant

$$\boldsymbol{X_o}: {}'a\ ordinal \to {}'a\ ordinal\ \mathbb{P}$$

$$\forall \beta \bullet\ X_o\ \beta = \{\eta \mid \eta <_o \beta\}$$

SML

```
val strong_infinity = new_axiom(["strong_infinity"], ⌜
∀β•
      ∃γ• β <o γ
∧
      ∀τ• τ <o γ ⇒
         ℙ (X o τ) <s X o γ
      ∧ (∀f• (∃ρ• (∀ν• ν <o τ ⇒ f ν <o ρ) ∧
                  (ρ ≤o γ ⇒ ρ <o γ)))

⌝);
```

Later the essential ideas here may be expressed in more conventional terms and used to validate this definition. Pro-tem, the following notes may illuminate the axiom.

The axiom is intended to state:

1. that every 'a ordinal is less than some inaccessible 'a ordinal
2. that the universe is the set of 'a ordinals less than some regular 'a ordinal

Thus γ in the axiom is the name used for this supposedly inaccessible 'a ordinal, but note that the least such γ will not be inaccessible, but will be ω, the first limit 'a ordinal. Adding the requirement that γ be uncountable does not strengthen the axiom which still entails that every 'a ordinal is less than some inaccessible 'a ordinal. What we assert of γ is first that it is a strong limit 'a ordinal and then that it (and the universe as a whole) is regular. These concepts are given formal definitions later, but the axiom is presented in concise form rather than through the definitions of the concepts.

It will be a while before any use is made of this axiom at all. For the meantime the elementary theorems obtained hold even in a singleton 'a ordinal type.

$|\boldsymbol{lt_o\_min\_cond} =$
$|\quad \vdash \forall\ A\bullet\ \neg\ A = \{\} \Rightarrow (\exists\ x\bullet\ x \in A \wedge (\forall\ y\bullet\ y \in A \Rightarrow \neg\ y <_o x))$
$|\boldsymbol{lt_o\_trans} =$
$|\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \wedge \gamma <_o \eta \Rightarrow \beta <_o \eta$
$|\boldsymbol{lt_o\_irrefl} =$
$|\quad \vdash \forall\ \beta\bullet\ \neg\ \beta <_o \beta$
$|\boldsymbol{lt_o\_trich} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta <_o \gamma \vee \gamma <_o \beta \vee \beta = \gamma$
$|\boldsymbol{lt_o\_trich\_fc} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \neg\ \beta <_o \gamma \wedge \neg\ \gamma <_o \beta \Rightarrow \beta = \gamma$
$|\boldsymbol{lt_o\_trich\_fc2} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \neg\ (\neg\ \beta <_o \gamma \wedge \neg\ \gamma <_o \beta \wedge \neg\ \beta = \gamma)$
$|\boldsymbol{\leqslant_o\_refl} =$
$|\quad \vdash \forall\ \beta\bullet\ \beta \leqslant_o \beta$
$|\boldsymbol{\leqslant_o\_lt_o} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \Leftrightarrow \neg\ \gamma <_o \beta$
$|\boldsymbol{\neg_o\_clauses} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ (\neg\ \beta <_o \gamma \Leftrightarrow \gamma \leqslant_o \beta) \wedge (\neg\ \gamma \leqslant_o \beta \Leftrightarrow \beta <_o \gamma)$

A useful principle for reasoning about the 'a ordinals is the following analogue of set theoretic extensionality:

$|\boldsymbol{ord\_ext\_thm} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta = \gamma \Leftrightarrow (\forall\ \delta\bullet\ \delta <_o \beta \Leftrightarrow \delta <_o \gamma)$

We we later make use of quasi extensional characterisations of operations over 'a ordinals, in which an 'a ordinal expression is characterised by a statement of the conditions under which 'a ordinals are less than the value of the expression. This facilitates proofs about 'a ordinals in which the complexity is on the right of an inequality, or in which such can be obtained by the extesionality principle above.

This leaves an awkwardness where our goal has an expression on the left of an inequality which the following rule is intended to ameliorate.

$|\boldsymbol{\leqslant_o\_ext\_thm} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \Leftrightarrow (\forall\ \delta\bullet\ \delta <_o \beta \Rightarrow \delta <_o \gamma)$

$|\boldsymbol{lt_o\_\leqslant_o} =$
$|\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \Rightarrow \beta \leqslant_o \gamma$
$|\boldsymbol{\leqslant_o\_trans} =$
$|\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \leqslant_o \gamma \wedge \gamma \leqslant_o \eta \Rightarrow \beta \leqslant_o \eta$
$|\boldsymbol{\leqslant_o\_lt_o\_trans} =$
$|\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \leqslant_o \gamma \wedge \gamma <_o \eta \Rightarrow \beta <_o \eta$
$|\boldsymbol{lt_o\_\leqslant_o\_trans} =$
$|\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \wedge \gamma \leqslant_o \eta \Rightarrow \beta <_o \eta$
$|\boldsymbol{\leqslant_o\_cases} =$
$|\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \vee \gamma \leqslant_o \beta$

It will be useful to have a name for the least element of a collection of 'a ordinals...

HOL Constant

$$\boldsymbol{Least_o}: 'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$$

$$\forall so\ \eta\bullet\ \eta \in so \Rightarrow Least_o\ so \in so\ \wedge\ Least_o\ so \leqslant_o \eta$$

... and for the supremum of a set of 'a ordinals.

HOL Constant

$$\boldsymbol{Ub_o}: 'a\ ordinal\ \mathbb{P} \to 'a\ ordinal\ \mathbb{P}$$

$$\forall so\bullet\ Ub_o\ so = \{\beta \mid \forall \eta\bullet\ \eta \in so \Rightarrow \eta \leqslant_o \beta\}$$

HOL Constant

$$\boldsymbol{SUb_o}: 'a\ ordinal\ \mathbb{P} \to 'a\ ordinal\ \mathbb{P}$$

$$\forall so\bullet\ SUb_o\ so = \{\beta \mid \forall \eta\bullet\ \eta \in so \Rightarrow \eta <_o \beta\}$$

HOL Constant

$$\boldsymbol{Sup_o}: 'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$$

$$\forall so\bullet\ Sup_o\ so = Least_o\ (Ub_o\ so)$$

HOL Constant

$$\boldsymbol{SSup_o}: 'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$$

$$\forall so\bullet\ SSup_o\ so = Least_o\ (SUb_o\ so)$$

$\boldsymbol{Least_o\_thm} =$

$$\vdash\ \forall\ so\ \beta\bullet\ \beta \in so \Rightarrow (\forall\ \gamma\bullet\ \gamma <_o Least_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \gamma <_o \eta))$$

$\boldsymbol{Ub_o\_thm} =$

$$\vdash\ \forall\ so\ \gamma\bullet\ \gamma \in Ub_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \eta \leqslant_o \gamma)$$

$\boldsymbol{Ub_o\_X_o\_thm} =$

$$\vdash\ \forall\ \alpha\bullet\ \alpha \in Ub_o\ (X_o\ \alpha)$$

$\boldsymbol{Ub_o\_X_o\_thm2} =$

$$\vdash\ \forall\ \alpha\bullet\ \alpha \in Ub_o\ \{\beta | \beta <_o \alpha\}$$

$\boldsymbol{SUb_o\_thm} =$

$$\vdash\ \forall\ so\ \gamma\bullet\ \gamma \in SUb_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \eta <_o \gamma)$$

$\boldsymbol{SUb_o\_X_o\_thm} =$

$$\vdash\ \forall\ \alpha\bullet\ \alpha \in SUb_o\ (X_o\ \alpha)$$

$\boldsymbol{SUb_o\_X_o\_thm2} =$

$$\vdash\ \forall\ \alpha\bullet\ \alpha \in SUb_o\ \{\beta | \beta <_o \alpha\}$$

$\boldsymbol{lt_o\_Sup_o} =$

$$\vdash\ \forall\ so\ \alpha\bullet\ \alpha \in Ub_o\ so \Rightarrow$$

$$(\forall \gamma\bullet\ \gamma <_o Sup_o\ so \Leftrightarrow (\exists \eta\bullet\ \eta \in so\ \wedge\ \gamma <_o \eta))$$

lt_o-Sup_{o2} =

$$\vdash\ \forall\ \alpha\ \gamma\bullet\ \gamma <_o Sup_o\ \{\beta | \beta <_o \alpha\} \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \alpha\ \wedge\ \gamma <_o \eta)$$

lt_o-$SSup_o$ =

$$\vdash\ \forall\ so\ \alpha\bullet\ \alpha \in SUb_o\ so \Rightarrow (\forall\ \gamma\bullet\ \gamma <_o SSup_o\ so \Leftrightarrow (\exists\ \eta\bullet\ \eta \in so\ \wedge\ \gamma \leqslant_o \eta))$$

$SSup_o$-lt_o =

$$\vdash\ \forall\ \alpha\bullet\ SSup_o\ \{\beta | \beta <_o \alpha\} = \alpha$$

$SSup_o$-lt_{o2} =

$$\vdash\ \forall\ so\ \alpha\ \beta\bullet\ \beta \in so\ \wedge\ \beta \in SUb_o\ so \Rightarrow (SSup_o\ so <_o \alpha \Leftrightarrow (\exists\ \beta\bullet\ \beta \in SUb_o\ so\ \wedge\ \beta <_o \alpha))$$

$SSup_o$-X_o =

$$\vdash\ \forall\ \alpha\bullet\ SSup_o\ (X_o\ \alpha) = \alpha$$

Now a name to the least 'a ordinal:

HOL Constant

0_o: *'a ordinal*

$$0_o = Least_o\ \{\delta\ |\ T\}$$

$zero_o$-thm =

$$\vdash\ \forall\ \beta\bullet\ 0_o \leqslant_o \beta$$

lt_o-$zero_o$-thm =

$$\vdash\ \forall\ \beta\bullet\ \neg\ \beta <_o 0_o$$

In order to define operators over the 'a ordinals (without undesirable complications) the 'a ordinals must be closed under the operations. If we want to use such operations in formulating our strong axiom of infinity, then we would need to assert sufficiently strong closure conditions in advance of our axiom of infinity.

The basis for the closure principle one which definitions of functions like 'a ordinal addition is based is a related to the axiom of replacement in set theory. In talking of 'a ordinals the corresponding notion is that or regularity, which we can define without any kind of axiom of infinity as follows.

First the notion of cofinality. This definition is perhaps a little eccentric, in that it is defined over all 'a ordinals not just limit 'a ordinals, and in that it is couched in terms of arbitrary functions rather than increasing sequences, and consequently takes the supremum of the image rather than the limit of a sequence.

The supremum of an image will prove more generally useful so we give it a name.

By the image of an 'a ordinal through a map, I mean the image of the set of 'a ordinals less than that 'a ordinal ():

HOL Constant

Image$_o$: $(('a\ ordinal \to 'b) \times 'a\ ordinal) \to 'b\ \mathbb{P}$

$$\forall f\ \beta\bullet\ Image_o(f, \beta) = \{\delta\ |\ \exists \eta\bullet\ \eta <_o \beta\ \wedge\ f\ \eta = \delta\}$$

$Image_o$-thm =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma \in Image_o\ (f,\ \beta) \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \beta \wedge \gamma = f\ \eta)$

$Image_o$-$zero_o$-thm =
$\vdash \forall\ f\bullet\ Image_o\ (f,\ 0_o) = \{\}$

$Image_o$-mono-thm =
$\vdash \forall\ f\ \alpha\ \beta\bullet\ \alpha \leqslant_o \beta \Rightarrow Image_o\ (f,\ \alpha) \subseteq Image_o\ (f,\ \beta)$

Ub_o-$Image_o$-thm =
$\vdash \forall\ f\ \beta\bullet\ \exists\ \gamma\bullet\ \gamma \in Ub_o\ (Image_o\ (f,\ \beta))$

Ub_o-$Image_o$-$zero_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma \in Ub_o\ (Image_o\ (f,\ 0_o))$

SUb_o-$Image_o$-thm =
$\vdash \forall\ f\ \beta\bullet\ \exists\ \gamma\bullet\ \gamma \in SUb_o\ (Image_o\ (f,\ \beta))$

SUb_o-$Image_o$-$zero_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma \in SUb_o\ (Image_o\ (f,\ 0_o))$

$SupIm_o$ is then the supremum of the image of an 'a ordinal. In the case that the function is increasing then this is the limit of a β sequence. Sometimes where such a limit is used in the literature there is no apparent benefit from the restriction to increasing sequences and I use $SupIm_o$ of an arbitary map, as in, for example, the definition of 'a ordinal addition.

HOL Constant

$SupIm_o$: $(('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal) \to 'a\ ordinal$

$\forall x\bullet\ SupIm_o\ x = Sup_o\ (Image_o\ x)$

$SSupIm_o$ is the strict supremum of the image of an 'a ordinal.

HOL Constant

$SSupIm_o$: $(('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal) \to 'a\ ordinal$

$\forall x\bullet\ SSupIm_o\ x = SSup_o\ (Image_o\ x)$

In general the supremum of an image only exists if the image is bounded above. However, one of the principle purposes of our axiom of strong infinity is to ensure that such bounds exist. By analogy with replacement in set theory, which tells us that the image of a set is a set, we know that the image of a bounded collection of 'a ordinals is itself bounded. This enables us to prove the following results about $SupIm_o$ and $SSupIm_o$.

lt_o-$SupIm_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma <_o SupIm_o\ (f,\ \beta) \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \beta \wedge \gamma <_o f\ \eta)$

$SupIm_o$-$zero_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \neg\ \gamma <_o SupIm_o\ (f,\ 0_o)$

lt_o-$SSupIm_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma <_o SSupIm_o\ (f,\ \beta) \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \beta \wedge \gamma \leqslant_o f\ \eta)$

$SSupIm_o$-$zero_o$ =
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \neg\ \gamma <_o SSupIm_o\ (f,\ 0_o)$

3.2 Defining Functions over the Ordinals

Often functions over *'a ordinals* are defined by cases in a manner analogous to primitive recursive definitions over the natural numbers (in which the cases are zero and successors) by adding a further case for limit 'a ordinals. Its not clear whether such an approach would work for us, because there is some difficulty in dealing with the limit case.

The approach I adopt addresses directly the limit case and subsumes the whole.

It may help to think of this as definition by inequality. Just as sets can be uniquely determined by identifying their members, so can 'a ordinals when they are represented by sets. Though we do not represent an 'a ordinal by a set, it is nevertheless uniquely determined by its predecessors, which would have been its members if we had been working in set theory.

Thus an 'a ordinal β might be defined by a sentence of the following form:

$$\forall\gamma\bullet\ \gamma <_o \beta \Leftrightarrow \mathit{formula}$$

I did look for a way of automatically proving the consistency of definitions in that form, but found this to be less straightforward than I had expected. The reason is that not all formulae of the given form are consistent. The formula on the right has to have the property that if true for a given value γ it is true also for any smaller value.

I have therefore to fall back on forms of definition more similar to those used in t042 [2].

Thus instead of the above we would have something like:

$$\beta = \mathit{SSup}_o\ \{\gamma \mid \mathit{formula}\}$$

Which is not subject to the same constraint.

A further problem is the necessary recursion in defining operations over 'a ordinals. A more definite example is desirable so we will look at addition.

Addition could be defined as follows:

$$\forall\beta\ \gamma\ \eta\bullet\ \eta <_o \beta +_o \gamma \Leftrightarrow \eta <_o \beta \vee \exists\rho\bullet\ \rho <_o \gamma \wedge \eta = \beta +_o \rho$$

The recursion here is well-founded because the addition on the right operates on smaller arguments than the one on the left. To make this conspicuous we can rewrite the definition, first:

$$\forall\beta\ \gamma\bullet\ \beta +_o \gamma = \mathit{SSup}_o\ \{\eta \mid \eta <_o \beta \vee \exists\rho\bullet\ \rho <_o \gamma \wedge \eta = \beta +_o \rho\}$$

This first step overcomes the first problem (the dependence on establishing that the formula 'downward closed', the set in the second formulation does not need to be downward closed). The smaller values become irrelevant, and this could be simplified to:

$$\forall\beta\ \gamma\bullet\ \beta +_o \gamma = \mathit{SSup}_o\ \{\eta \mid \exists\rho\bullet\ \rho <_o \gamma \wedge \eta = \beta +_o \rho\}$$

A further step allows the well-foundedness of the recursion to be made more obvious.

$$\forall\beta\ \gamma\bullet\ (\$+_o\ \beta)\ \gamma = \mathit{SSup}_o\ (\mathit{Image}_o\ (\$+_o\ \beta)\ \gamma)$$

It is a feature of $SSupIm_o(\$ +_o\beta)\gamma$ that it accessed values of $\$ +_o\beta$ only for 'a ordinals less than γ. A suitable recursion theorem is necessary to permit definitions in this form to be accepted.

There is a question in formulating such a recursion theorem as to what access to the function is required. A maximally general theorem will allow access to a restricted version of the function, an intermediate version to the image of the values below some 'a ordinal through the map, and a minimal version to the supremum of strict supremum of the values. At this point it is not clear to me what is likely to be most useful.

On considering this I came to the conclusion that a general principle should be provided elsewhere, and have put one (tf_rec_thm2) in t009 [1]. This provides a recursion theorem for use with any well-founded relation.

When we specialise that to the ordering over the 'a ordinals we get:

$$\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet f\ x = af\ ((x, \$<_o) \langle\lhd f)\ x$$

In which the operator $\langle \lhd$ restricts the domain of function f hiding information about values for arguments not lower in the ordering than x. This can be made a little slicker for use in this document by defining a more specific restriction operator:

SML

```
declare_infix(400, "◁o");
```

HOL Constant

$$\$\lhd_o : 'a\ ordinal \to ('a\ ordinal \to 'b) \to ('a\ ordinal \to 'b)$$

$$\forall x\ f \bullet x \lhd_o f = (x, \$<_o) \langle\lhd f$$

$$\vdash \forall\gamma\ f\ \beta \bullet \beta <_o \gamma \Rightarrow (\gamma \lhd_o f)\ \beta = f\ \beta$$

$$\vdash \forall\ \gamma\ f \bullet Image_o\ (\gamma \lhd_o f, \gamma) = Image_o\ (f, \gamma)$$

$$\vdash \forall\ \gamma\ f \bullet SupIm_o\ (\gamma \lhd_o f, \gamma) = SupIm_o\ (f, \gamma)$$

$$\vdash \forall\gamma\ f \bullet SSupIm_o\ (\gamma \lhd_o f, \gamma) = SSupIm_o\ (f, \gamma)$$

Hence:

$$\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet f\ x = af\ (x \lhd_o f)\ x$$

Unfortunately this will not work with the ProofPower consistency prover, which requires a constructor (as if we were defining a function by pattern matching over a recursive data type). To get automatic consistency proofs we need to add a dummy constructor, so:

$$\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet f\ (CombI\ x) = af\ (x \lhd_o f)\ x$$

$$\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet f\ (CombI\ x) = af\ (Image_o\ (f, x))\ x$$

Rather than having specific recursion theorems for definitions involving SupIm or SSupIm, we apply the required domain restriction to the function being defined wherever it is used on the right of the definition.

SML

```
force_new_pc "'ordcard-rec1";
add_∃_cd_thms [ord_rec_thm3] "'ordcard-rec1";
```

3.3 Ordinal Arithmetic

SML

```
declare_infix(400, "+o");
```

The sum of two 'a ordinals is the strict supremum of the set of 'a ordinals less than the second operand under the function which adds each 'a ordinal to the first operand.

HOL Constant

$$\$+_o : 'a\ ordinal \to 'a\ ordinal \to 'a\ ordinal$$

$$\forall \beta\ \gamma \bullet\ \beta +_o \gamma = if\ \gamma = 0_o\ then\ \beta\ else\ SupIm_o\ (\$+_o\ \beta, \gamma)$$

$$\boldsymbol{plus_o\_0_o} =$$
$$\vdash\ \forall\ \beta \bullet\ \beta +_o 0_o = \beta$$

3.4 Defining Inaccessibility

The significance of this section to the purposes of this work is moot, since the strong axiom of infinity, which implicitly asserts the existence of inaccessible 'a ordinals, does not depend upon an explicit definition.

The purpose of this section is therefore as a kind of check on the formulation of that axiom. This check could go as far as defining inaccessible and proving the equivalence of the give axiom with a formulation based on the defined concept. However, to serve that pupose this material would have to come before the axiom, since in the context of that axiom we cannot distinguish between equivalence and entailment of the new formulation by the old.

Co-finality is usually a relation between increasing β sequences (β a limit 'a ordinal) and some limit 'a ordinal α. I don't yet have sequences, so its convenient to give a slightly broader definition. Instead of increasing sequences I allow the image of any 'a ordinal under a function (which need not be increasing). At this point I don't actually understand why an increasing sequence is

Such an image is "cofinal" in an 'a ordinal if:

- the image falls entirely below the 'a ordinal
- the supremum of the image is that 'a ordinal

SML

```
declare_infix(400, "CofinalIn_o");
```

HOL Constant

$\$\boldsymbol{CofinalIn_o}$: $(('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal) \to 'a\ ordinal \to BOOL$

$\forall x\ \gamma\bullet\ x\ CofinalIn_o\ \gamma \Leftrightarrow Image_o\ x \subseteq X_o\ \gamma \wedge SupIm_o\ x = \gamma$

HOL Constant

$\boldsymbol{Cf_o}$: $'a\ ordinal \to 'a\ ordinal$

$\forall\beta\bullet\ Cf_o\ \beta = Least_o\ \{\gamma \mid \exists f\bullet\ (f, \gamma)\ CofinalIn_o\ \beta\}$

We can now define the notion of regularity, one of the defining properties of inaccessible cardinals.

HOL Constant

$\boldsymbol{Regular_o}$: $'a\ ordinal \to BOOL$

$\forall\beta\bullet\ Regular_o\ \beta \Leftrightarrow Cf_o\ \beta = \beta$

HOL Constant

$\boldsymbol{Singular_o}$: $'a\ ordinal \to BOOL$

$\forall\beta\bullet\ Singular_o\ \beta \Leftrightarrow \neg\ Regular_o\ \beta$

As well as using this in the definition of inaccessibility we want to be able to state that the universe is regular (to get sufficiently generous recursion principles, analogous to global replacement). The vocabulary above doesn't really help in stating this global principle, but it is simple enough to state directly. We will come to that later.

To get inaccessibilty we need also to express the notion of a strong limit.

HOL Constant

$\boldsymbol{Succ_o}$: $'a\ ordinal \to 'a\ ordinal$

$\forall\beta\bullet\ Succ_o\ \beta = Least_o\ \{\gamma \mid \beta <_o \gamma\}$

HOL Constant

$\boldsymbol{Successor_o}$: $'a\ ordinal \to BOOL$

$\forall\beta\bullet\ Successor_o\ \beta \Leftrightarrow \exists\gamma\bullet\ \beta = Succ_o\ \gamma$

HOL Constant

$\boldsymbol{Limit_o}$: $'a\ ordinal \to BOOL$

$\forall\beta\bullet\ Limit_o\ \beta \Leftrightarrow 0_o <_o \beta \wedge \neg\ Successor_o\ \beta$

HOL Constant

$\boldsymbol{\omega_o}$: $'a\ ordinal$

$\omega_o = Least_o\ \{\beta \mid Limit_o\ \beta\}$

HOL Constant

$$\boldsymbol{StrongLimit_o}: 'a\ ordinal \rightarrow BOOL$$

$$\forall\beta\bullet\ StrongLimit_o\ \beta \Leftrightarrow \forall\gamma\bullet\ \gamma <_o \beta \Rightarrow \mathbb{P}\ (X_o\ \gamma) <_s X_o\ \beta$$

SML

```
val strong_infinity2 = ⌜
∀β•  (∃γ• β <o γ ∧
            Regular o γ ∧
            StrongLimit o γ)
   ∧
      (∀f• (∃ρ• (∀ν• ν <o β ⇒ f ν <o ρ)))
⌝;
```

The basic idea is to state that every 'a ordinal is less than some (strongly) inaccessible 'a ordinal (also a cardinal), with a little tweak to give, in effect, global replacement (or its analogue for a theory of 'a ordinals rather than sets). Here local replacement is the requirement that each 'a ordinal is less than some regular cardinal, global replacement tells us that the universe is regular. The other requirement is that this regular cardinal is a strong limit, i.e. closed under powerset.

To validate this principle I could first prove the principal in the set theory in t023, or alternatively in t041 since the 'a ordinals are further developed there. That would gives greater confidence in its consistency. That it is adequate can be testing in use, or by constructing a set theory from this type of 'a ordinals which satisifies the first principle.

However, without further validation I now proceed to establish that it can be used to justify a convenient recursion principle.

The first step in this is to define a couple of functions using our axiom of infinity.

The first is a function which give an infinite 'a ordinal will deliver the least inaccessible 'a ordinal greater than that 'a ordinal, given a finite 'a ordinal it returns ω. I will call this *Lio*.

HOL Constant

$$\boldsymbol{G_o}: 'a\ ordinal \rightarrow 'a\ ordinal$$

$$\forall\beta\bullet\ G_o\ \beta = Least_o\ \{\gamma \mid \beta <_o \gamma \wedge \omega_o <_o \gamma$$
$$\wedge\ \forall\tau\bullet\ \tau <_o \gamma \Rightarrow$$
$$\mathbb{P}\ (X_o\ \tau) <_s X_o\ \gamma$$
$$\wedge\ (\forall f\bullet\ (\forall\nu\bullet\ \nu <_o \tau \Rightarrow f\ \nu <_o \tau)$$
$$\Rightarrow\ (\exists\rho\bullet\ \rho <_o \gamma \wedge (\forall\nu\bullet\ \nu <_o \tau \Rightarrow f\ \nu <_o \rho)))\}$$

4 CARDINALS

Its by no means clear that a type of cardinals is necessary for my purposes, so in the first instance the development of this type will be quite limited. I have put it in just to see whether it proves useful rather than in a firm expectation that it will. In fact I put in the type of cardinals before I came up

with the strong infinity axiom above which does not make use of the cardinals, and my first thought about how the type might be useful was to think it might make possibly a neat strong axiom. The one I have now is pretty neat, but possibly it might look less of a Kluge if I talked about cardinality using the cardinals rather that by using cardinality comparisons over ProofPower sets.

4.1 The Type of Cardinals

One could introduce cardinals in a manner similar to the introduction of 'a ordinals, but we would then have no coupling between the two types. We want the cardinals to be the initial 'a ordinals, so that a strong axiom of infinity for the 'a ordinals populates the cardinals as well.

With that in mind we need the vocabulary to talk about initial 'a ordinals before we can set up a type of cardinals.

$$
\begin{array}{|l}
\boldsymbol{lt_{o}\text{-}\subseteq} = \\
\quad \vdash \forall\ \beta\ \gamma\bullet\ \gamma <_o \beta \Rightarrow X_o\ \gamma \subseteq X_o\ \beta \\
\boldsymbol{lt_{o}\text{-}\subset} = \\
\quad \vdash \forall\ \beta\ \gamma\bullet\ \gamma <_o \beta \Rightarrow X_o\ \gamma \subset X_o\ \beta \\
\boldsymbol{\leqslant_{o}\text{-}\subseteq} = \\
\quad \vdash \forall \beta\ \gamma\bullet\ \gamma \leqslant_o \beta \Rightarrow X_o\ \gamma \subseteq X_o\ \beta
\end{array}
$$

The ordering of 'a ordinals by cardinality is then:

SML

$$
\left|\ declare\_infix(300,\ \texttt{"}\leqslant_{oc}\texttt{"});\right.
$$

HOL Constant

$$
\begin{array}{|l}
\$\boldsymbol{\leqslant_{oc}}\colon\ 'a\ ordinal \rightarrow 'a\ ordinal \rightarrow BOOL \\
\hline
\forall \beta\ \gamma\bullet\ \beta \leqslant_{oc} \gamma \Leftrightarrow X_o\ \beta \leqslant_s X_o\ \gamma
\end{array}
$$

$$
\begin{array}{|l}
\boldsymbol{\leqslant_{oc}\text{-}refl} = \\
\quad \vdash \forall\ \beta\bullet\ \beta \leqslant_{oc} \beta \\
\boldsymbol{lt_{o}\text{-}\leqslant_{oc}} = \\
\quad \vdash \forall\ \beta\ \gamma\bullet\ \gamma <_o \beta \Rightarrow \gamma \leqslant_{oc} \beta \\
\boldsymbol{\leqslant_{oc}\text{-}trans} = \\
\quad \vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \leqslant_{oc} \gamma \wedge \gamma \leqslant_{oc} \eta \Rightarrow \beta \leqslant_{oc} \eta \\
\boldsymbol{\leqslant_{oc}\text{-}cases} = \\
\quad \vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_{oc} \gamma \vee \gamma \leqslant_{oc} \beta
\end{array}
$$

SML

$$
\left|\ declare\_infix(300,\ \texttt{"}<_{oc}\texttt{"});\right.
$$

HOL Constant

$$
\begin{array}{|l}
\$\boldsymbol{<_{oc}}\colon\ 'a\ ordinal \rightarrow 'a\ ordinal \rightarrow BOOL \\
\hline
\forall \beta\ \gamma\bullet\ \beta <_{oc} \gamma \Leftrightarrow \neg\ \gamma \leqslant_{oc} \beta
\end{array}
$$

$<_{oc}\_\mathbf{irrefl}$ =
$\vdash \forall\ \beta\bullet\ \neg\ \beta <_{oc} \beta$
$\leqslant_{oc}\_\neg\_\mathbf{lt_{oc}}$ =
$\vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_{oc} \gamma \Rightarrow \neg\ \gamma <_{oc} \beta$

SML
declare_infix(300, "$\sim_{oc}$");

HOL Constant
$\$\sim_{\mathbf{oc}}$: *'a ordinal* → *'a ordinal* → *BOOL*

$\forall\beta\ \gamma\bullet\ \beta \sim_{oc} \gamma \Leftrightarrow \beta \leqslant_{oc} \gamma \wedge \gamma \leqslant_{oc} \beta$

$\leqslant_{oc}\_\mathbf{cases2}$ =
$\vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_{oc} \gamma \Leftrightarrow \beta <_{oc} \gamma \vee \beta \sim_{oc} \gamma$
$\sim_{oc}\_\mathbf{refl}$ =
$\vdash \forall\ \beta\bullet\ \beta \sim_{oc} \beta$
$\mathbf{eq_{oc}}\_\mathbf{sym}$ =
$\vdash \forall\ \beta\ \gamma\bullet\ \beta \sim_{oc} \gamma \Rightarrow \gamma \sim_{oc} \beta$
$\mathbf{eq_{oc}}\_\mathbf{trans}$ =
$\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \sim_{oc} \gamma \wedge \gamma \sim_{oc} \eta \Rightarrow \beta \sim_{oc} \eta$
$\mathbf{lt_{oc}}\_\mathbf{trich}$ =
$\vdash \forall\ \beta\ \gamma\bullet\ \beta <_{oc} \gamma \vee \gamma <_{oc} \beta \vee \beta \sim_{oc} \gamma$

We have to define the notion of initiality. A initial 'a ordinal is one which is not smaller than or equal in cardinality with any smaller 'a ordinal.

HOL Constant
$\mathbf{InitialOrdinal}$: *'a ordinal* → *BOOL*

$\forall\beta\bullet\ InitialOrdinal\ \beta \Leftrightarrow \forall\gamma\bullet\ \gamma <_o \beta \Rightarrow \gamma <_{oc} \beta$

Before introducing a type using this predicate we must prove that there exists an initial 'a ordinal, for which the witness is the least 'a ordinal, obtainable using the minimal condition.

$\mathbf{InitialOrdinal}\_\mathbf{exists}$ =
$\vdash \exists\ \beta\bullet\ InitialOrdinal\ \beta$

$\mathbf{InitialOrdinals}\_\mathbf{exist}$ =
$\vdash \forall\ \beta\bullet \exists\ \delta\bullet\ InitialOrdinal\ \delta \wedge \delta \sim_{oc} \beta$

$\mathbf{InitialOrdinal}\_\mathbf{eq}$ =
$\vdash \forall\ \beta\ \gamma\bullet\ InitialOrdinal\ \beta \wedge InitialOrdinal\ \gamma \wedge \beta \sim_{oc} \gamma \Rightarrow \beta = \gamma$

Now we can introduce a new type represented by the initial 'a ordinals.

SML

```
val cardinal_def =
      new_type_defn(["cardinal"], "cardinal", ["'a"], InitialOrdinal_exists);
```

There are various functions between the 'a ordinals and cardinals which may be used in formulating a strong axiom of infinity. The type definition defines the new type as having the same cardinality as the initial 'a ordinals, and we use this bijection to determine the correspondence between cardinals and their alephs. The abstraction function would normally be determined only over the alephs, but it will be more useful to have an abstraction function which yields the cardinality of any 'a ordinal.

These two maps can be defined thus:

HOL Constant

$$\boldsymbol{Ord_c}: {}'a\ cardinal \to {}'a\ ordinal;$$
$$\boldsymbol{Card_o}: {}'a\ ordinal \to {}'a\ cardinal$$

$$\begin{aligned}
&(\forall \beta{:}'a\ cardinal\bullet\ Card_o(Ord_c\ \beta) = \beta)\\
&\wedge\ (\forall \beta{:}'a\ ordinal\bullet\ InitialOrdinal\ \beta \Leftrightarrow Ord_c\ (Card_o\ \beta) = \beta)\\
&\wedge\ OneOne\ Ord_c\\
&\wedge\ (\forall\ \beta\bullet\ InitialOrdinal\ (Ord_c\ \beta))\\
&\wedge\ (\forall \beta\ \gamma\bullet\ \beta \sim_{oc} \gamma \Rightarrow Card_o\ \beta = Card_o\ \gamma)
\end{aligned}$$

5 INFINITARY SEQUENCES

Infinitary sequences are functions whose domain is an 'a ordinal. To make a type of them we will need to use HOL total functions over the type of 'a ordinals, and the domain would then be fixed. We therefore use an ordered pair consisting of an 'a ordinal (which is the domain) together with a total function over the 'a ordinals. The values of this function outside the domain are immaterial, but the fact that the function has such values confuses the identity conditions and we must take steps to ensure that the identity conditions come out right. We could either ensure that in the new type all functions take the same value everywhere outside the domain, or else we could use an equivalence class of functions which take the same values over the domain.

I don't know which of these two would be simplest; I shall plump for the first since it is more familiar to me.

The following predicate determines the representatives of infinitary sequences.

HOL Constant

$$\boldsymbol{ISeqRep}: {}'a\ ordinal \times ({}'a\ ordinal \to {}'b) \to BOOL$$

$$\forall p\bullet\ ISeqRep\ p \Leftrightarrow \forall or\bullet \neg\ or <_o Fst\ p \Rightarrow Snd\ p\ or = \epsilon x\bullet T$$

We now define destructor/constructor operations over these sequences.

HOL Constant

$$\boldsymbol{MkISeq}: {'a}\ ordinal \times ({'a}\ ordinal \to {'b}) \to ({'a},{'b})iseq;$$
$$\boldsymbol{DestISeq}: ({'a},{'b})iseq \to {'a}\ ordinal \times ({'a}\ ordinal \to {'b})$$

$$(\forall \beta\ \gamma \bullet\ DestISeq\ \beta = DestISeq\ \gamma \Rightarrow \beta = \gamma)$$
$$\wedge \quad (\forall \beta \bullet\ ISeqRep\ (DestISeq\ \beta))$$
$$\wedge \quad (\forall \beta \bullet\ MkISeq\ (DestISeq\ \beta) = \beta)$$
$$\wedge \quad (\forall p \bullet\ ISeqRep\ p \Rightarrow DestISeq(MkISeq\ p) = p)$$

HOL Constant

$$\boldsymbol{Length_{is}}: ({'a},{'b})iseq \to {'a}\ ordinal$$

$$\forall is \bullet\ Length_{is}\ is = Fst\ (DestISeq\ is)$$

HOL Constant

$$\boldsymbol{Function_{is}}: ({'a},{'b})iseq \to ({'a}\ ordinal \to {'b})$$

$$\forall is \bullet\ Function_{is}\ is = Snd\ (DestISeq\ is)$$

HOL Constant

$$\boldsymbol{Elems_{is}}: ({'a},{'b})iseq \to {'b}\ \mathbb{P}$$

$$\forall is \bullet\ Elems_{is}\ is = \{e \mid \exists \beta \bullet\ \beta <_o (Length_{is}\ is)$$
$$\wedge\ e = (Function_{is}\ is)\ \beta\}$$

6 INFINITARY TREES

An infinitary tree is to be represented by a partial function from sequences of 'a ordinals to some type of labels. The sequences are the coordinates of nodes in the tree, and the labels label each node. There is a well-formedness condition which ensures that the set of coordinates of branches from any node is an initial segment of the 'a ordinals.

HOL Constant

$$\boldsymbol{ITreeRep}: ({'a}\ ordinal\ LIST \to {'a} + ONE) \to BOOL$$

$$\forall f \bullet\ ITreeRep\ f \Leftrightarrow$$
$$\forall l{:}{'a}\ ordinal\ LIST \bullet\ \exists \beta \bullet\ \{\gamma \mid IsL\ (f\ (l\ @\ [\gamma]))\} = X_o\ \beta$$

$$\boldsymbol{iTree\_def} = \vdash\ \exists\ f \bullet\ TypeDefn\ ITreeRep\ f$$

We now define a constructor/destructor pair for this new type of object.

6.0.1 Proof Context

SML
```
add_pc_thms "'ordcard" ([]);
add_rw_thms [] "'ordcard";
add_sc_thms [] "'ordcard";

set_merge_pcs ["basic_hol", "'ordcard"];
commit_pc "'ordcard";
```

6.1 Closing

7 TRANSFINTE DATATYPES FROM ORDINALS

My original idea for the work in this document arose from questioning whether an axiomatisation of a set theory was the best way to strengthen HOL, and from wondering whether, for example, a type of infinitary trees might serve the kinds of purpose for which I have until now uses axiomatic set theory in a neater way.

More recently, reflections on how to extract the power from a strong axiom of infinity (of which the axiom for the *'a ordinals* above is an example), I have wondered whether the kinds of desired construction could not be done directly from the *'a ordinals* without an intermediate construction of a type of infinitary trees.

A first exploration of ways of exploiting such an axiom of infinity may be undertaken through the exercise of constructing from the 'a ordinals the largest possible initial segment of the cumulative heirarchy. I spent a little time thinking about this case, eventually arriving at some ideas for a solution to the more general problem of obtaining representation for mutually recursive infinitary datatypes.

Thinking about this has lead me to an augmentation to the basis on which the axiom of infinity is expressed. I originally took a new type and chose an arbitrary well-ordering of the type. I now think it would be better to chose a minimal well-ordering, i.e. one which is an initial 'a ordinal. This will give better closure properties in the resulting initial segment of the cumulative heirarchy. However, I don't need to make that change before attempting the construction, which is what I aim to do here. To support polymorphic structures I have concluded that the axiom of infinity should be stated for a polymorphic rather than a monomorphic type. The resulting infinity is then required to exceed the cardinality of the type parameter. This is necessary to make instantiation of a polymorphic datatype to a large type (such as ⌜:ONE ordinal⌝) work

The method is to give a definition by transfinite induction/recursion of am enumeration of the values to be obtained by projection from the new abstract types. This enumeration is then the projection function for the new type (a combined projector for all the constructors of whatever type delivering for a single type definition the disjoint union of the domain types of the constructors, for multiple types a disjoint union of disjoint unions). Of course, if there are multiple types there could not be a combined function from those types, it would not be well typed. The function goes not from the abstract types themselves, but from their common representation type, the ordinals, which are partitioned to give distinct representations for the different types being introduced. This single projection function can then be used to define separate projection functions for each of the abstract types introduced.

7.1 Definitions for the General Construction

A single operator is to be defined which takes a parameter characterising the required new types. This characterisation is provided by a function which, given an infinitary sequence of values, returns the set of values which can be constructed from those values. The domain of this function will be a partial version of the desired uniform projector (a total function over the type of indexes, the ordinals, together with an index indicating the domain over which that function is to be considered defined, the index is the strict supremum of the ordinals over which the function is considered defined), so that the projection (so far) can be used to test the types of the available components, and the indexes can be used in constructing new values. The enumeration of all the values is constructed in order of rank (i.e. the number of constructions necessary to obtain the desired value from nothing). Within each rank a least well-ordering is obtained by use of the choice function. New values are selected for coding by taking the next element from a least well-ordering obtained by choice of the values of the same rank. When this well-ordering is exhausted a new rank will be coded, if there are any new values which can be obtained by applying the functor again to the new collection of codings.

To make the recursion work in HOL we define an auxiliary function which delivers extra information so that we can remember the set of uncoded elements at the current rank until it has been exhausted.

The information supplied by the auxiliary function for each 'a ordinal is as follows:

1. the value assigned to this 'a ordinal
2. the rank of the value coded by this 'a ordinal
3. the set of new values at this rank yet to be coded
4. a well-ordering of the not yet coded values at this rank
5. if this is a valid code then T else F (this will be false if all possible values have already been assigned a code)

The final coding is projected from this function.

The question which faces us here is, given an initial segment of a map from 'a ordinals to structures of the above kind, how we deliver the next element of the map. Stating it informally, first we check whether the coding has been completed, by testing for a code (ordinal) below the present value which has not been assigned a value. If there is such a code the present code will have no value either. Otherwise we establish whether the new value will be of the same rank as its latest predecessors, or whether it will be the first of the next rank.

To determine this we first check whether there is a highest rank among the predecessors. If not, then the next value coded must have a limit 'a ordinal as its rank, and must be the first value of that rank.

If there is a highest rank among the predecessors we then form the set of values at that rank which have not yet been coded, by taking the intersection of the set of uncoded values associated with each coded value at this rank. If this set is empty then the current 'a ordinal must be the first to code a set of the next higher rank, otherwise the 'a ordinal will code a set of the same rank as its latest predecessors and its position in that rank will be the strict supremum of all the positions occupied by its predecessors at that rank. The value it codes is obtained using the choice function on the set of as yet uncoded values at that rank.

If we are starting a new rank then the new rank is the strict supremum of the previous ranks. The first 'a ordinal coding a value of that rank is the present 'a ordinal. We obtain the set of values at

this rank by taking the set of all those values constructable from previous 'a ordinals which are not already coded by one of them. We then chose an element of that collection, which will be the value coded by the present 'a ordinal, and remove that element from the set of as-yet uncoded values.

On the basis of this sketch I propose to define separately the test for an 'a ordinal coding the first set of a rank, the values of the 4-tuple in that case and the value of the 4 tuple otherwise before combining these in an inductive definition.

HOL Labelled Product

5TUP

$TValue$: $'b$;
$TRank$: $'a\ ordinal$;
$TRes$: $'b\ \mathbb{P}$;
TWo: $'b \to 'b \to BOOL$;
$TValid$: $BOOL$

The ranks which have been partly or wholly coded can be extracted from a partial enumeration as follows:

HOL Constant

$$\boldsymbol{Ranks}: (('a\ ordinal \to ('a, 'b)5TUP) \times 'a\ ordinal) \to 'a\ ordinal\ \mathbb{P}$$

$$\forall f\ x\bullet\ Ranks\ (f, x) = \{or \mid \exists z\bullet\ TRank\ (f\ z) = or\}$$

The first rank which has not yet been partially or wholly coded is the strict supremum of those ranks

HOL Constant

$$\boldsymbol{SSRank}: (('a\ ordinal \to ('a, 'b)5TUP) \times 'a\ ordinal) \to 'a\ ordinal$$

$$\forall f\ x\bullet\ SSRank\ (f, x) = SSup_o\ (Ranks\ (f, x))$$

If SSRank is a limit ordinal then the next coded element will be of that new rank. Otherwise we need to know whether the predecessor rank is exhausted. In that case the predecessor is the supremum.

HOL Constant

$$\boldsymbol{SRank}: (('a\ ordinal \to ('a, 'b)5TUP) \times 'a\ ordinal) \to 'a\ ordinal$$

$$\forall f\ x\bullet\ SRank\ (f, x) = Sup_o\ (Ranks\ (f, x))$$

To chose another element of some incomplete rank for coding we need to get the set of not-yet coded elements. This is the set of elements each of which is in the residual set for every coded element of that rank (or the intersection of those residues).

HOL Constant

$$\boldsymbol{RankRes1}: (('a\ ordinal \to ('a, 'b)5TUP) \times 'a\ ordinal) \times 'a\ ordinal \to 'b\ \mathbb{P}$$

$$\forall f\ x\ y\bullet\ RankRes1\ ((f, x), y) = \bigcap \{r \mid \exists ro\bullet\ ro <_o x \wedge TRank(f\ ro) = y \wedge r = TRes(f\ ro)\}$$

This function delivers the residual set, which may be empty, without being told the rank.

HOL Constant

$$\textbf{\textit{RankRes2}}: ((\prime a\ ordinal \to (\prime a, \prime b)5TUP) \times \prime a\ ordinal) \to \prime b\ \mathbb{P}$$

$$\forall f\ x \bullet\ RankRes2\ (f, x) =$$
$$\quad if\ Limit_o\ (SSRank\ (f, x))$$
$$\quad then\ \{\}$$
$$\quad else\ RankRes1\ ((f, x), SRank\ (f, x))$$

If there are no values to be coded then we need to step up to the next rank. To do this we collect together all the values so far coded, apply the constructor function to determine what can be constructed from them, and remove from the reaults anything which has already been coded.

The following function obtains the set of values already coded.

HOL Constant

$$\textbf{\textit{ValuesCoded}}: ((\prime a\ ordinal \to (\prime a, \prime b)5TUP) \times \prime a\ ordinal) \to \prime b\ SET$$

$$\forall f\ x \bullet\ ValuesCoded\ (f, x) = \{y \mid \exists z \bullet\ z <_o x \wedge TValid(f\ z) \wedge TValue(f\ z) = y\}$$

This functor upgrades the constructor function to one which only returns new values.

HOL Constant

$$\textbf{\textit{NewValFunc}}: (((\prime a\ ordinal \to (\prime a, \prime b)5TUP) \times \prime a\ ordinal) \to \prime b\ \mathbb{P}) \to (((\prime a\ ordinal \to (\prime a, \prime b)5TUP) \times \prime a\ ordinal) \to \prime b\ \mathbb{P})$$

$$\forall m \bullet\ NewValFunc\ m = \lambda s \bullet\ (m\ s) \setminus (ValuesCoded\ s)$$

The following function takes an initial segment (the predecessors of some 'a ordinal) of the map from articles to 5TUPs and computes the next 5TUP.

The prior enumeration of 5TUPs, the new rank, and the function for computing the elements of this rank are supplied as arguments.

HOL Constant

$$\textbf{\textit{Next4T}}: (\prime a\ ordinal \to (\prime a, \prime b)5TUP) \to \prime a\ ordinal \to (((\prime a\ ordinal \to (\prime a, \prime b)5TUP) \times \prime a\ ordinal) \to \prime b\ SET) \to (\prime a, \prime b)5TUP$$

$$\forall (f{:}\prime a\ ordinal \to (\prime a, \prime b)5TUP)\ x\ m \bullet$$
$$\ Next4T\ f\ x\ m =$$
$$\quad if\ \exists z{:}\prime a\ ordinal \bullet\ z <_o x \wedge \neg\ TValid(f\ z)$$
$$\quad then\ Mk5TUP\ (\epsilon x{:}\prime b \bullet T)\ (\epsilon x{:}\prime a\ ordinal \bullet T)\ (\epsilon x{:}\prime b\ \mathbb{P} \bullet T)$$
$$\qquad\qquad (\epsilon x{:}\prime b \to \prime b \to BOOL \bullet T)\ F$$
$$\quad else\ \ let\ res = RankRes2\ (f, x)$$
$$\qquad in\ \ \ if\ res = \{\}$$
$$\qquad\quad then\ \ let\ nr = SSRank\ (f,x)$$

```
                    and nvs = (NewValFunc m) (f, x)
                    in
    if nvs = {}
    then Mk5TUP (ϵx:'b• T) (ϵx:'a ordinal• T) (ϵx:'b ℙ• T)
                        (ϵx:'b → 'b → BOOL• T) F
    else  let nv = ϵx:'b• x ∈ nvs (* need to chose least *)
          in    Mk5TUP nv nr (nvs \ {nv}) (ϵx:'b → 'b → BOOL• T) T

                else  let nv = ϵx:'b• x ∈ res
                      in Mk5TUP nv (SRank (f, x)) (res \ {nv}) (ϵx:'b → 'b → BOOL• T) T
```

Now we use the above function in a definition by transfinite recursion.

SML

```
push_merge_pcs ["ordcard0", "'ordcard", "'ordcard-rec1"];


set_goal([], ⌜∃Map2Coding:(((('a ordinal → ('a, 'b)5TUP) × 'a ordinal) → 'b SET) → ('a ordinal → ('a
          ∀(m:(((('a ordinal → ('a, 'b)5TUP) × 'a ordinal) → 'b SET)) x•
    Map2Coding m x = Next4T (x ◁o (Map2Coding m)) x m⌝);
a (prove_∃_tac);
a (strip_tac);
a (LEMMA_T ⌜∃Map2Coding':('a ordinal → ('a, 'b)5TUP)•
          ∀x•
    Map2Coding' (CombI x) = Next4T (x ◁o Map2Coding') x m'⌝
    (accept_tac o (pure_rewrite_rule [get_spec ⌜CombI⌝]))
    THEN1 basic_prove_∃_tac);
val Map2Coding_consistent = save_cs_∃_thm (pop_thm());

pop_pc();
```

HOL Constant

```
Map2Coding: ((('a ordinal → ('a, 'b)5TUP) × 'a ordinal)  → 'b SET) → ('a ordinal → ('a, 'b)5TU
─────────────────────────────────────────────────────────────────────
∀m x•  Map2Coding m x = Next4T (x ◁o (Map2Coding m)) x m
```

The following function extracts the projection function from the result of this operation.

HOL Constant

```
Coding2Projection: ('a ordinal → ('a, 'b)5TUP) → ('a ordinal → 'b)
─────────────────────────────────────────────────────────────────────
∀c x•  Coding2Projection c x = TValue (c x)
```

7.2 Some Theorems

The development of the theory here is somewhat ad. hoc., driven by the needs of the examples which follow.

A general pattern should emerge which is similar to the kinds of results noramlly obtained when recursive datatype are introduced, with certain modifications arising from the infinitary nature of the facility.

However, in this development the exceptions are the norm, and the results normally expected may not be derived early, (or at all). Much of the complication in the non-infinitary cases arises from the presence of multiple types and of multiple constructors for each type. In the simplest infinitary example, which is where we start here (set theory), there is only one constructor, set formation, and only one type, the type of pure sets.

In that case the key results required are firstly that the projection is a bijection, and a principal of induction on the rank of the construction (or on an ordering correponding to the order of the ordinal codes).

7.2.1 That Projections are Bijections

An elementary requirement is that the projection function is a bijection over its domain of well-definedness (which will in the examples usually be the whole type of ordinals, though if the constructions are used for more ordinary datatypes will not always be the case).

7.3 Set Theory from Ordinals

This section uses the simplest of transfinite inductive definitions to check the definitions and force the development of the general theory above.

We define a membership relation over the 'a ordinals which should make it into an initial segment of the cumulative hierarchy. In this example its not strictly necessary to introduce a new type of sets, since the membership relation will be defined over the entire type of ;a ordinals.

The polymorphism can be handled in two ways. In the more complex treatment, a set of ordinals having the same cardinality as the type parameter are taken as urelemente, and the construction carries on from there.

In the simpler case we simply ignore the type parameter, begin by enumerating the power set of the empty set and take it from there. The effect of type parameterisation is solely in its influence on how far you can go, as a result of its effect on the cardinality of the ordinals.

Since the motivation for the type parameter was the second of these, and this is also the easiest to do, thats what I will use as the first test case.

7.3.1 Defining the Membership Relation

I now define the required map function, which takes an encoding of a collection of sets, and returns the set of sets which can be constructed from that collection (its power set).

HOL Constant

$$\begin{array}{l} \textbf{\textit{SetsMap}}: ((' a \ ordinal \to (' a, ' a \ ordinal \ SET) 5TUP) \times ' a \ ordinal) \\ \qquad \to ' a \ ordinal \ SET \ SET \\ \hline \forall f \ x \bullet \ \ SetsMap \ (f, \ x) = \mathbb{P} \ (X_o \ x) \end{array}$$

Using the resulting projection function we can define membership over the ordinals as follows:

SML

$declare\_infix(230,"\in_o");$

HOL Constant

$\$\in_{\boldsymbol{o}}: 'a\ ordinal \to 'a\ ordinal \to BOOL$

$\forall x\ y \bullet\ \ x \in_o y \Leftrightarrow x \in (Coding2Projection\ (Map2Coding\ SetsMap)\ y)$

7.3.2 Extensionality

The first thing to prove is extensionality. Extensionality is the consequence in this simple example of the general feature of this kind of construction, that the projection function is a bijection (hence no two ordinals code the same set of ordinals. So before presenting the extensionality theorem I step back to prove that the codings are all bijections (over their defined part).

References

[1] R.D. Arthan and R.B. Jones. Well-orderings and Well-foundedness. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t009.pdf`.

[2] Roger Bishop Jones. A Higher Order Theory of Well-Founded Sets (with Urelements). *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t042.pdf`.

A The Theory ordcard0

A.1 Parents

wf_recp *wf_relp* *trees* *U_orders* *rbjmisc*

A.2 Children

ordcard

A.3 Constants

$\$\leqslant_s$ $'a\ \mathbb{P} \to 'b\ \mathbb{P} \to BOOL$
$\$<_s$ $'a\ \mathbb{P} \to 'b\ \mathbb{P} \to BOOL$
$\$\sim_s$ $'a\ \mathbb{P} \to 'b\ \mathbb{P} \to BOOL$

A.4 Fixity

Right Infix 300:

$<_s$ $\sim_s$ $\leqslant_s$

A.5 Definitions

$\leqslant_s$
$\vdash \forall\ A\ B$
$\bullet\ A \leqslant_s B$
$\Leftrightarrow (\exists\ f$
$\bullet \forall\ x\ y$
$\bullet\ x \in A \wedge y \in A$
$\Rightarrow f\ x \in B \wedge f\ y \in B \wedge (f\ x = f\ y \Rightarrow x = y))$

$<_s$
$\vdash \forall\ A\ B \bullet\ A <_s B \Leftrightarrow A \leqslant_s B \wedge \neg\ B \leqslant_s A$

$\sim_s$
$\vdash \forall\ A\ B$
$\bullet\ A \sim_s B$
$\Leftrightarrow (\exists\ f\ g$
$\bullet (\forall\ x \bullet\ x \in A \Rightarrow f\ x \in B \wedge g\ (f\ x) = x)$
$\wedge (\forall\ y \bullet\ y \in B \Rightarrow g\ y \in A \wedge f\ (g\ y) = y))$

A.6 Theorems

$\leqslant_s$***refl***
$\vdash \forall\ A \bullet\ A \leqslant_s A$

$\subseteq$***_*** $\leqslant_s$***thm***
$\vdash \forall\ A\ B \bullet\ A \subseteq B \Rightarrow A \leqslant_s B$

$\leqslant_s$***trans***
$\vdash \forall\ A\ B\ C \bullet\ A \leqslant_s B \wedge B \leqslant_s C \Rightarrow A \leqslant_s C$

lt$_s$***irrefl***
$\vdash \forall\ A \bullet \neg\ A <_s A$

lt$_s$***trans***
$\vdash \forall\ A\ B\ C \bullet\ A <_s B \wedge B <_s C \Rightarrow A <_s C$

lt$_s$***_***$\leqslant_s$***trans***
$\vdash \forall\ A\ B\ C \bullet\ A <_s B \wedge B \leqslant_s C \Rightarrow A <_s C$

$\leqslant_s$***lt***$_s$***trans***
$\vdash \forall\ A\ B\ C \bullet\ A \leqslant_s B \wedge B <_s C \Rightarrow A <_s C$

card_equiv_lemma
$\vdash \forall\ x\ y\ z$
$\bullet\ x \sim_s x$
$\wedge (x \sim_s y \Leftrightarrow y \sim_s x)$
$\wedge (x \sim_s y \wedge y \sim_s z \Rightarrow x \sim_s z)$

B The Theory ordcard

B.1 Parents

wf_recp *wf_relp* *U_orders* *ordcard0*

B.2 Constants

$\$<_o$	$'a\ ordinal \to 'a\ ordinal \to BOOL$
$\$\leqslant_o$	$'a\ ordinal \to 'a\ ordinal \to BOOL$
$\boldsymbol{X_o}$	$'a\ ordinal \to 'a\ ordinal\ \mathbb{P}$
$\boldsymbol{Least_o}$	$'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$
$\boldsymbol{Ub_o}$	$'a\ ordinal\ \mathbb{P} \to 'a\ ordinal\ \mathbb{P}$
$\boldsymbol{SUb_o}$	$'a\ ordinal\ \mathbb{P} \to 'a\ ordinal\ \mathbb{P}$
$\boldsymbol{Sup_o}$	$'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$
$\boldsymbol{SSup_o}$	$'a\ ordinal\ \mathbb{P} \to 'a\ ordinal$
$\mathbf{0_o}$	$'a\ ordinal$
$\boldsymbol{Image_o}$	$('a\ ordinal \to 'b) \times 'a\ ordinal \to 'b\ \mathbb{P}$
$\boldsymbol{SupIm_o}$	$('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal \to 'a\ ordinal$
$\boldsymbol{SSupIm_o}$	$('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal \to 'a\ ordinal$
$\$\lhd_o$	$'a\ ordinal \to ('a\ ordinal \to 'b) \to 'a\ ordinal \to 'b$
$\$+_o$	$'a\ ordinal \to 'a\ ordinal \to 'a\ ordinal$
$\$\boldsymbol{CofinalIn_o}$	$('a\ ordinal \to 'a\ ordinal) \times 'a\ ordinal$ $\to 'a\ ordinal$ $\to BOOL$
$\boldsymbol{Cf_o}$	$'a\ ordinal \to 'a\ ordinal$
$\boldsymbol{Regular_o}$	$'a\ ordinal \to BOOL$
$\boldsymbol{Singular_o}$	$'a\ ordinal \to BOOL$
$\boldsymbol{Succ_o}$	$'a\ ordinal \to 'a\ ordinal$
$\boldsymbol{Successor_o}$	$'a\ ordinal \to BOOL$
$\boldsymbol{Limit_o}$	$'a\ ordinal \to BOOL$
$\boldsymbol{\omega_o}$	$'a\ ordinal$
$\boldsymbol{StrongLimit_o}$	$'a\ ordinal \to BOOL$
$\boldsymbol{G_o}$	$'a\ ordinal \to 'a\ ordinal$
$\$\leqslant_{oc}$	$'a\ ordinal \to 'a\ ordinal \to BOOL$
$\$<_{oc}$	$'a\ ordinal \to 'a\ ordinal \to BOOL$
$\$\sim_{oc}$	$'a\ ordinal \to 'a\ ordinal \to BOOL$
$\boldsymbol{InitialOrdinal}$	$'a\ ordinal \to BOOL$
$\boldsymbol{Card_o}$	$'a\ ordinal \to 'a\ cardinal$
$\boldsymbol{Ord_c}$	$'a\ cardinal \to 'a\ ordinal$
$\boldsymbol{ISeqRep}$	$'a\ ordinal \times ('a\ ordinal \to 'b) \to BOOL$
$\boldsymbol{DestISeq}$	$('a, 'b)\ iseq \to 'a\ ordinal \times ('a\ ordinal \to 'b)$
$\boldsymbol{MkISeq}$	$'a\ ordinal \times ('a\ ordinal \to 'b) \to ('a, 'b)\ iseq$
$\boldsymbol{Length_{is}}$	$('a, 'b)\ iseq \to 'a\ ordinal$
$\boldsymbol{Function_{is}}$	$('a, 'b)\ iseq \to 'a\ ordinal \to 'b$
$\boldsymbol{Elems_{is}}$	$('a, 'b)\ iseq \to 'b\ \mathbb{P}$
$\boldsymbol{ITreeRep}$	$('a\ ordinal\ LIST \to 'a + ONE) \to BOOL$
$\boldsymbol{TValid}$	$('a, 'b)\ 5TUP \to BOOL$
$\boldsymbol{TWo}$	$('a, 'b)\ 5TUP \to 'b \to 'b \to BOOL$

$\boldsymbol{TRes}$ $('a, 'b)\ 5TUP \to 'b\ \mathbb{P}$
$\boldsymbol{TRank}$ $('a, 'b)\ 5TUP \to 'a\ ordinal$
$\boldsymbol{TValue}$ $('a, 'b)\ 5TUP \to 'b$
$\boldsymbol{Mk5TUP}$ $'b \to 'a\ ordinal \to 'b\ \mathbb{P} \to ('b \to 'b \to BOOL) \to BOOL \to ('a, 'b)\ 5TUP$
$\boldsymbol{Ranks}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'a\ ordinal\ \mathbb{P}$
$\boldsymbol{SSRank}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'a\ ordinal$
$\boldsymbol{SRank}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'a\ ordinal$
$\boldsymbol{RankRes1}$ $(('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal) \times 'a\ ordinal \to 'b\ \mathbb{P}$
$\boldsymbol{RankRes2}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}$
$\boldsymbol{ValuesCoded}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}$
$\boldsymbol{NewValFunc}$ $(('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}) \to ('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}$
$\boldsymbol{Next4T}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \to 'a\ ordinal \to (('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}) \to ('a, 'b)\ 5TUP$
$\boldsymbol{Map2Coding}$ $(('a\ ordinal \to ('a, 'b)\ 5TUP) \times 'a\ ordinal \to 'b\ \mathbb{P}) \to 'a\ ordinal \to ('a, 'b)\ 5TUP$
$\boldsymbol{Coding2Projection}$ $('a\ ordinal \to ('a, 'b)\ 5TUP) \to 'a\ ordinal \to 'b$
$\boldsymbol{SetsMap}$ $('a\ ordinal \to ('a, 'a\ ordinal\ \mathbb{P})\ 5TUP) \times 'a\ ordinal \to 'a\ ordinal\ \mathbb{P}\ \mathbb{P}$
$\$\in_{\boldsymbol{o}}$ $'a\ ordinal \to 'a\ ordinal \to BOOL$

B.3 Types

$'1\ \boldsymbol{ordinal}$
$'1\ \boldsymbol{cardinal}$
$('1, '2)\ \boldsymbol{iseq}$
$'1\ \boldsymbol{iTree}$
$('1, '2)\ \boldsymbol{5TUP}$

B.4 Fixity

Right Infix 230:
$\in_{\boldsymbol{o}}$
Right Infix 300:
$<_{\boldsymbol{o}}$ $<_{\boldsymbol{oc}}$ $\sim_{\boldsymbol{oc}}$ $\leqslant_{\boldsymbol{o}}$ $\leqslant_{\boldsymbol{oc}}$
Right Infix 400:
$\boldsymbol{CofinalIn}_{\boldsymbol{o}}$ $+_{\boldsymbol{o}}$ $\lhd_{\boldsymbol{o}}$

B.5 Axioms

card_slb $\vdash$ *Universe* $<_s$ *Universe*

strong_infinity

$$\begin{aligned}
&\vdash \forall\ \beta \\
&\quad \bullet\ \exists\ \gamma \\
&\qquad \bullet\ \beta <_o \gamma \\
&\qquad\quad \wedge\ (\forall\ \tau \\
&\qquad\quad \bullet\ \tau <_o \gamma \\
&\qquad\qquad \Rightarrow \mathbb{P}\ (X_o\ \tau) <_s X_o\ \gamma \\
&\qquad\qquad\quad \wedge\ (\forall\ f \\
&\qquad\qquad\quad \bullet\ \exists\ \rho \\
&\qquad\qquad\qquad \bullet\ (\forall\ \nu \bullet\ \nu <_o \tau \Rightarrow f\ \nu <_o \rho) \\
&\qquad\qquad\qquad\quad \wedge\ (\rho \leqslant_o \gamma \Rightarrow \rho <_o \gamma)))
\end{aligned}$$

B.6 Definitions

$<_o$ $\vdash$ *WellOrdering* (*Universe*, \$$<_o$)
$\wedge$ *WellFounded* (*Universe*, \$$<_o$)

$\leqslant_o$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta \leqslant_o \gamma \Leftrightarrow \beta <_o \gamma \vee \beta = \gamma$

X_o $\vdash \forall\ \beta \bullet\ X_o\ \beta = \{\eta | \eta <_o \beta\}$

$Least_o$ $\vdash \forall\ so\ \eta \bullet\ \eta \in so \Rightarrow Least_o\ so \in so \wedge Least_o\ so \leqslant_o \eta$

Ub_o $\vdash \forall\ so \bullet\ Ub_o\ so = \{\beta | \forall\ \eta \bullet\ \eta \in so \Rightarrow \eta \leqslant_o \beta\}$

SUb_o $\vdash \forall\ so \bullet\ SUb_o\ so = \{\beta | \forall\ \eta \bullet\ \eta \in so \Rightarrow \eta <_o \beta\}$

Sup_o $\vdash \forall\ so \bullet\ Sup_o\ so = Least_o\ (Ub_o\ so)$

$SSup_o$ $\vdash \forall\ so \bullet\ SSup_o\ so = Least_o\ (SUb_o\ so)$

0_o $\vdash 0_o = Least_o\ \{\delta | T\}$

$Image_o$ $\vdash \forall\ f\ \beta \bullet\ Image_o\ (f, \beta) = \{\delta | \exists\ \eta \bullet\ \eta <_o \beta \wedge f\ \eta = \delta\}$

$SupIm_o$ $\vdash \forall\ x \bullet\ SupIm_o\ x = Sup_o\ (Image_o\ x)$

$SSupIm_o$ $\vdash \forall\ x \bullet\ SSupIm_o\ x = SSup_o\ (Image_o\ x)$

$\lhd_o$ $\vdash \forall\ x\ f \bullet\ x \lhd_o f = (x, \$<_o) \langle \lhd f$

$+_o$ $\vdash \forall\ \beta\ \gamma$
$\bullet\ \beta +_o \gamma$
$= (if\ \gamma = 0_o\ then\ \beta\ else\ SupIm_o\ (\$+_o\ \beta, \gamma))$

$CofinalIn_o$ $\vdash \forall\ x\ \gamma$
$\bullet\ x\ CofinalIn_o\ \gamma \Leftrightarrow Image_o\ x \subseteq X_o\ \gamma \wedge SupIm_o\ x = \gamma$

Cf_o $\vdash \forall\ \beta \bullet\ Cf_o\ \beta = Least_o\ \{\gamma | \exists\ f \bullet\ (f, \gamma)\ CofinalIn_o\ \beta\}$

$Regular_o$ $\vdash \forall\ \beta \bullet\ Regular_o\ \beta \Leftrightarrow Cf_o\ \beta = \beta$

$Singular_o$ $\vdash \forall\ \beta \bullet\ Singular_o\ \beta \Leftrightarrow \neg\ Regular_o\ \beta$

$Succ_o$ $\vdash \forall\ \beta \bullet\ Succ_o\ \beta = Least_o\ \{\gamma | \beta <_o \gamma\}$

$Successor_o$ $\vdash \forall\ \beta \bullet\ Successor_o\ \beta \Leftrightarrow (\exists\ \gamma \bullet\ \beta = Succ_o\ \gamma)$

$Limit_o$ $\vdash \forall\ \beta \bullet\ Limit_o\ \beta \Leftrightarrow 0_o <_o \beta \wedge \neg\ Successor_o\ \beta$

ω_o $\vdash \omega_o = Least_o\ \{\beta | Limit_o\ \beta\}$

$StrongLimit_o$
$\vdash \forall\ \beta$
$\bullet\ StrongLimit_o\ \beta$
$\Leftrightarrow (\forall\ \gamma \bullet\ \gamma <_o \beta \Rightarrow \mathbb{P}\ (X_o\ \gamma) <_s X_o\ \beta)$

G_o $\vdash \forall\ \beta$
$\bullet\ G_o\ \beta$
$= Least_o$
$\{\gamma$

$|\beta <_o \gamma$
$\land\ \omega_o <_o \gamma$
$\land\ (\forall\ \tau$
$\bullet\ \tau <_o \gamma$
$\Rightarrow \mathbb{P}\ (X_o\ \tau) <_s X_o\ \gamma$
$\land\ (\forall\ f$
$\bullet\ (\forall\ \nu \bullet\ \nu <_o \tau \Rightarrow f\ \nu <_o \tau)$
$\Rightarrow (\exists\ \rho$
$\bullet\ \rho <_o \gamma$
$\land\ (\forall\ \nu$
$\bullet\ \nu <_o \tau \Rightarrow f\ \nu <_o \rho))))\}$

$\boldsymbol{\leqslant_{oc}}$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta \leqslant_{oc} \gamma \Leftrightarrow X_o\ \beta \leqslant_s X_o\ \gamma$
$\boldsymbol{<_{oc}}$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta <_{oc} \gamma \Leftrightarrow \neg\ \gamma \leqslant_{oc} \beta$
$\boldsymbol{\sim_{oc}}$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta \sim_{oc} \gamma \Leftrightarrow \beta \leqslant_{oc} \gamma \land \gamma \leqslant_{oc} \beta$
InitialOrdinal
$\vdash \forall\ \beta \bullet\ \mathit{InitialOrdinal}\ \beta \Leftrightarrow (\forall\ \gamma \bullet\ \gamma <_o \beta \Rightarrow \gamma <_{oc} \beta)$
cardinal $\vdash \exists\ f \bullet\ \mathit{TypeDefn}\ \mathit{InitialOrdinal}\ f$
$\boldsymbol{Ord_c}$
$\boldsymbol{Card_o}$ $\vdash (\forall\ \beta \bullet\ \mathit{Card}_o\ (\mathit{Ord}_c\ \beta) = \beta)$
$\land\ (\forall\ \beta \bullet\ \mathit{InitialOrdinal}\ \beta \Leftrightarrow \mathit{Ord}_c\ (\mathit{Card}_o\ \beta) = \beta)$
$\land\ \mathit{OneOne}\ \mathit{Ord}_c$
$\land\ (\forall\ \beta \bullet\ \mathit{InitialOrdinal}\ (\mathit{Ord}_c\ \beta))$
$\land\ (\forall\ \beta\ \gamma \bullet\ \beta \sim_{oc} \gamma \Rightarrow \mathit{Card}_o\ \beta = \mathit{Card}_o\ \gamma)$
ISeqRep $\vdash \forall\ p$
$\bullet\ \mathit{ISeqRep}\ p$
$\Leftrightarrow (\forall\ or \bullet\ \neg\ or <_o \mathit{Fst}\ p \Rightarrow \mathit{Snd}\ p\ or = (\epsilon\ x \bullet\ T))$
iseq $\vdash \exists\ f \bullet\ \mathit{TypeDefn}\ \mathit{ISeqRep}\ f$
MkISeq
DestISeq $\vdash (\forall\ \beta\ \gamma \bullet\ \mathit{DestISeq}\ \beta = \mathit{DestISeq}\ \gamma \Rightarrow \beta = \gamma)$
$\land\ (\forall\ \beta \bullet\ \mathit{ISeqRep}\ (\mathit{DestISeq}\ \beta))$
$\land\ (\forall\ \beta \bullet\ \mathit{MkISeq}\ (\mathit{DestISeq}\ \beta) = \beta)$
$\land\ (\forall\ p \bullet\ \mathit{ISeqRep}\ p \Rightarrow \mathit{DestISeq}\ (\mathit{MkISeq}\ p) = p)$
$\boldsymbol{Length_{is}}$ $\vdash \forall\ is \bullet\ \mathit{Length}_{is}\ is = \mathit{Fst}\ (\mathit{DestISeq}\ is)$
$\boldsymbol{Function_{is}}$ $\vdash \forall\ is \bullet\ \mathit{Function}_{is}\ is = \mathit{Snd}\ (\mathit{DestISeq}\ is)$
$\boldsymbol{Elems_{is}}$ $\vdash \forall\ is$
$\bullet\ \mathit{Elems}_{is}\ is$
$= \{e$
$|\exists\ \beta$
$\bullet\ \beta <_o \mathit{Length}_{is}\ is \land e = \mathit{Function}_{is}\ is\ \beta\}$
ITreeRep $\vdash \forall\ f$
$\bullet\ \mathit{ITreeRep}\ f$
$\Leftrightarrow (\forall\ l \bullet\ \exists\ \beta \bullet\ \{\gamma | \mathit{IsL}\ (f\ (l\ @\ [\gamma]))\} = X_o\ \beta)$
iTree $\vdash \exists\ f \bullet\ \mathit{TypeDefn}\ \mathit{ITreeRep}\ f$
5TUP $\vdash \exists\ f \bullet\ \mathit{TypeDefn}\ (\lambda\ x \bullet\ T)\ f$
Mk5TUP
TValue
TRank
TRes
TWo
TValid $\vdash \forall\ t\ x1\ x2\ x3\ x4\ x5$
$\bullet\ \mathit{TValue}\ (\mathit{Mk5TUP}\ x1\ x2\ x3\ x4\ x5) = x1$

$$
\begin{array}{l}
\quad \wedge\ TRank\ (Mk5TUP\ x1\ x2\ x3\ x4\ x5) = x2 \\
\quad \wedge\ TRes\ (Mk5TUP\ x1\ x2\ x3\ x4\ x5) = x3 \\
\quad \wedge\ TWo\ (Mk5TUP\ x1\ x2\ x3\ x4\ x5) = x4 \\
\quad \wedge\ (TValid\ (Mk5TUP\ x1\ x2\ x3\ x4\ x5) \Leftrightarrow x5) \\
\quad \wedge\ Mk5TUP \\
\qquad (TValue\ t) \\
\qquad (TRank\ t) \\
\qquad (TRes\ t) \\
\qquad (TWo\ t) \\
\qquad (TValid\ t) \\
\quad = t
\end{array}
$$

$$
\begin{array}{ll}
\textbf{\textit{Ranks}} & \vdash \forall\ f\ x\bullet\ Ranks\ (f,\ x) = \{or | \exists\ z\bullet\ TRank\ (f\ z) = or\} \\
\textbf{\textit{SSRank}} & \vdash \forall\ f\ x\bullet\ SSRank\ (f,\ x) = SSup_o\ (Ranks\ (f,\ x)) \\
\textbf{\textit{SRank}} & \vdash \forall\ f\ x\bullet\ SRank\ (f,\ x) = Sup_o\ (Ranks\ (f,\ x)) \\
\textbf{\textit{RankRes1}} & \vdash \forall\ f\ x\ y \\
& \quad \bullet\ RankRes1\ ((f,\ x),\ y) \\
& \qquad = \bigcap \\
& \qquad \{r \\
& \qquad\quad | \exists\ ro \\
& \qquad\qquad \bullet\ ro <_o x \\
& \qquad\qquad\quad \wedge\ TRank\ (f\ ro) = y \\
& \qquad\qquad\quad \wedge\ r = TRes\ (f\ ro)\} \\
\textbf{\textit{RankRes2}} & \vdash \forall\ f\ x \\
& \quad \bullet\ RankRes2\ (f,\ x) \\
& \qquad = (if\ Limit_o\ (SSRank\ (f,\ x)) \\
& \qquad\ then\ \{\} \\
& \qquad\ else\ RankRes1\ ((f,\ x),\ SRank\ (f,\ x))) \\
\textbf{\textit{ValuesCoded}} & \vdash \forall\ f\ x \\
& \quad \bullet\ ValuesCoded\ (f,\ x) \\
& \qquad = \{y \\
& \qquad | \exists\ z\bullet\ z <_o x \wedge TValid\ (f\ z) \wedge TValue\ (f\ z) = y\} \\
\textbf{\textit{NewValFunc}} & \vdash \forall\ m\bullet\ NewValFunc\ m = (\lambda\ s\bullet\ m\ s \setminus ValuesCoded\ s) \\
\textbf{\textit{Next4T}} & \vdash \forall\ f\ x\ m \\
& \quad \bullet\ Next4T\ f\ x\ m \\
& \qquad = (if\ \exists\ z\bullet\ z <_o x \wedge \neg\ TValid\ (f\ z) \\
& \qquad\ then \\
& \qquad\quad Mk5TUP\ (\epsilon\ x\bullet\ T)\ (\epsilon\ x\bullet\ T)\ (\epsilon\ x\bullet\ T)\ (\epsilon\ x\bullet\ T)\ F \\
& \qquad\ else \\
& \qquad\quad (let\ res = RankRes2\ (f,\ x) \\
& \qquad\qquad in\ if\ res = \{\} \\
& \qquad\qquad then \\
& \qquad\qquad\quad let\ nr = SSRank\ (f,\ x) \\
& \qquad\qquad\qquad and\ nvs = NewValFunc\ m\ (f,\ x) \\
& \qquad\qquad\qquad in\ if\ nvs = \{\} \\
& \qquad\qquad\qquad then \\
& \qquad\qquad\qquad\quad Mk5TUP \\
& \qquad\qquad\qquad\qquad (\epsilon\ x\bullet\ T) \\
& \qquad\qquad\qquad\qquad (\epsilon\ x\bullet\ T) \\
& \qquad\qquad\qquad\qquad (\epsilon\ x\bullet\ T) \\
& \qquad\qquad\qquad\qquad (\epsilon\ x\bullet\ T) \\
& \qquad\qquad\qquad\qquad F
\end{array}
$$

$else$
$(let\ nv = (\epsilon\ x\bullet\ x \in nvs)$
$in\ Mk5TUP$
nv
nr
$(nvs \setminus \{nv\})$
$(\epsilon\ x\bullet\ T)$
$T)$
$else$
$(let\ nv = (\epsilon\ x\bullet\ x \in res)$
$in\ Mk5TUP$
nv
$(SRank\ (f,\ x))$
$(res \setminus \{nv\})$
$(\epsilon\ x\bullet\ T)$
$T)))$

Map2Coding $\vdash \forall\ m\ x$
$\bullet\ Map2Coding\ m\ x = Next4T\ (x \lhd_o Map2Coding\ m)\ x\ m$

Coding2Projection
$\vdash \forall\ c\ x\bullet\ Coding2Projection\ c\ x = TValue\ (c\ x)$

SetsMap $\vdash \forall\ f\ x\bullet\ SetsMap\ (f,\ x) = \mathbb{P}\ (X_o\ x)$

$\in_o$ $\vdash \forall\ x\ y$
$\bullet\ x \in_o y$
$\Leftrightarrow x \in Coding2Projection\ (Map2Coding\ SetsMap)\ y$

B.7 Theorems

lt$_o$_wf $\vdash well\_founded\ \$<_o$

lt$_o$_min_cond
$\vdash \forall\ A$
$\bullet\ \neg\ A = \{\} \Rightarrow (\exists\ x\bullet\ x \in A \land (\forall\ y\bullet\ y \in A \Rightarrow \neg\ y <_o x))$

lt$_o$_trans $\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \land \gamma <_o \eta \Rightarrow \beta <_o \eta$

lt$_o$_irrefl $\vdash \forall\ \beta\bullet\ \neg\ \beta <_o \beta$

lt$_o$_trich $\vdash \forall\ \beta\ \gamma\bullet\ \beta <_o \gamma \lor \gamma <_o \beta \lor \beta = \gamma$

lt$_o$_trich_fc
$\vdash \forall\ \beta\ \gamma\bullet\ \neg\ \beta <_o \gamma \land \neg\ \gamma <_o \beta \Rightarrow \beta = \gamma$

lt$_o$_trich_fc2
$\vdash \forall\ \beta\ \gamma\bullet\ \neg\ (\neg\ \beta <_o \gamma \land \neg\ \gamma <_o \beta \land \neg\ \beta = \gamma)$

$\leqslant_o$_refl $\vdash \forall\ \beta\bullet\ \beta \leqslant_o \beta$

$\leqslant_o$_lt$_o$ $\vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \Leftrightarrow \neg\ \gamma <_o \beta$

$\neg_o$_clauses $\vdash \forall\ \beta\ \gamma\bullet\ (\neg\ \beta <_o \gamma \Leftrightarrow \gamma \leqslant_o \beta) \land (\neg\ \gamma \leqslant_o \beta \Leftrightarrow \beta <_o \gamma)$

ord_ext_thm $\vdash \forall\ \beta\ \gamma\bullet\ \beta = \gamma \Leftrightarrow (\forall\ \delta\bullet\ \delta <_o \beta \Leftrightarrow \delta <_o \gamma)$

lt$_o$_$\leqslant_o$ $\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \Rightarrow \beta \leqslant_o \gamma$

$\leqslant_o$_trans $\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \leqslant_o \gamma \land \gamma \leqslant_o \eta \Rightarrow \beta \leqslant_o \eta$

$\leqslant_o$_lt$_o$_trans
$\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta \leqslant_o \gamma \land \gamma <_o \eta \Rightarrow \beta <_o \eta$

lt$_o$_$\leqslant_o$_trans
$\vdash \forall\ \beta\ \gamma\ \eta\bullet\ \beta <_o \gamma \land \gamma \leqslant_o \eta \Rightarrow \beta <_o \eta$

$\leqslant_o$_cases $\vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \lor \gamma \leqslant_o \beta$

$\leqslant_o$_ext_thm $\vdash \forall\ \beta\ \gamma\bullet\ \beta \leqslant_o \gamma \Leftrightarrow (\forall\ \delta\bullet\ \delta <_o \beta \Rightarrow \delta <_o \gamma)$

Least$_o$_thm $\vdash \forall\ so\ \beta$

$\bullet\ \beta \in so$
$\Rightarrow (\forall\ \gamma$
$\bullet\ \gamma <_o Least_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \gamma <_o \eta))$

Ub_o_thm $\vdash \forall\ so\ \gamma\bullet\ \gamma \in Ub_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \eta \leqslant_o \gamma)$

Ub_o_X_o_thm $\vdash \forall\ \alpha\bullet\ \alpha \in Ub_o\ (X_o\ \alpha)$

Ub_o_X_o_thm2
$\vdash \forall\ \alpha\bullet\ \alpha \in Ub_o\ \{\beta | \beta <_o \alpha\}$

SUb_o_thm $\vdash \forall\ so\ \gamma\bullet\ \gamma \in SUb_o\ so \Leftrightarrow (\forall\ \eta\bullet\ \eta \in so \Rightarrow \eta <_o \gamma)$

SUb_o_X_o_thm
$\vdash \forall\ \alpha\bullet\ \alpha \in SUb_o\ (X_o\ \alpha)$

SUb_o_X_o_thm2
$\vdash \forall\ \alpha\bullet\ \alpha \in SUb_o\ \{\beta | \beta <_o \alpha\}$

lt_o_Sup_o $\vdash \forall\ so\ \alpha$
$\bullet\ \alpha \in Ub_o\ so$
$\Rightarrow (\forall\ \gamma\bullet\ \gamma <_o Sup_o\ so \Leftrightarrow (\exists\ \eta\bullet\ \eta \in so\ \wedge\ \gamma <_o \eta))$

lt_o_Sup_o2 $\vdash \forall\ \alpha\ \gamma$
$\bullet\ \gamma <_o Sup_o\ \{\beta | \beta <_o \alpha\} \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \alpha\ \wedge\ \gamma <_o \eta)$

lt_o_$SSup_o$ $\vdash \forall\ so\ \alpha$
$\bullet\ \alpha \in SUb_o\ so$
$\Rightarrow (\forall\ \gamma$
$\bullet\ \gamma <_o SSup_o\ so \Leftrightarrow (\exists\ \eta\bullet\ \eta \in so\ \wedge\ \gamma \leqslant_o \eta))$

$SSup_o$_lt_o $\vdash \forall\ \alpha\bullet\ SSup_o\ \{\beta | \beta <_o \alpha\} = \alpha$

$SSup_o$_X_o $\vdash \forall\ \alpha\bullet\ SSup_o\ (X_o\ \alpha) = \alpha$

$SSup_o$_lt_o2 $\vdash \forall\ so\ \beta\ \gamma$
$\bullet\ \beta \in so\ \wedge\ \gamma \in SUb_o\ so$
$\Rightarrow (\forall\ \alpha$
$\bullet\ SSup_o\ so <_o \alpha$
$\Leftrightarrow (\exists\ \eta\bullet\ \eta \in SUb_o\ so\ \wedge\ \eta <_o \alpha))$

$zero_o$_thm $\vdash \forall\ \beta\bullet\ 0_o \leqslant_o \beta$

lt_o_$zero_o$_thm
$\vdash \forall\ \beta\bullet\ \neg\ \beta <_o 0_o$

$Image_o$_thm $\vdash \forall\ f\ \beta\ \gamma$
$\bullet\ \gamma \in Image_o\ (f, \beta) \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \beta\ \wedge\ \gamma = f\ \eta)$

$Image_o$_$zero_o$_thm
$\vdash \forall\ f\bullet\ Image_o\ (f, 0_o) = \{\}$

$Image_o$_mono_thm
$\vdash \forall\ f\ \alpha\ \beta\bullet\ \alpha \leqslant_o \beta \Rightarrow Image_o\ (f, \alpha) \subseteq Image_o\ (f, \beta)$

Ub_o_$Image_o$_thm
$\vdash \forall\ f\ \beta\bullet\ \exists\ \gamma\bullet\ \gamma \in Ub_o\ (Image_o\ (f, \beta))$

Ub_o_$Image_o$_$zero_o$
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma \in Ub_o\ (Image_o\ (f, 0_o))$

SUb_o_$Image_o$_thm
$\vdash \forall\ f\ \beta\bullet\ \exists\ \gamma\bullet\ \gamma \in SUb_o\ (Image_o\ (f, \beta))$

SUb_o_$Image_o$_$zero_o$
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \gamma \in SUb_o\ (Image_o\ (f, 0_o))$

lt_o_$SupIm_o$ $\vdash \forall\ f\ \beta\ \gamma$
$\bullet\ \gamma <_o SupIm_o\ (f, \beta) \Leftrightarrow (\exists\ \eta\bullet\ \eta <_o \beta\ \wedge\ \gamma <_o f\ \eta)$

$SupIm_o$_$zero_o$
$\vdash \forall\ f\ \beta\ \gamma\bullet\ \neg\ \gamma <_o SupIm_o\ (f, 0_o)$

lt_o_$SSupIm_o$
$\vdash \forall\ f\ \beta\ \gamma$

$\bullet\ \gamma <_o SSupIm_o\ (f,\ \beta) \Leftrightarrow (\exists\ \eta \bullet\ \eta <_o \beta \wedge \gamma \leqslant_o f\ \eta)$

$SSupIm_o$-$zero_o$
$\vdash \forall\ f \bullet\ SSupIm_o\ (f,\ 0_o) = 0_o$

$\lhd_o$-fc $\vdash \forall\ \gamma\ f\ \beta \bullet\ \beta <_o \gamma \Rightarrow (\gamma \lhd_o f)\ \beta = f\ \beta$

$Image_o$-$\lhd_o$-thm
$\vdash \forall\ \gamma\ f \bullet\ Image_o\ (\gamma \lhd_o f,\ \gamma) = Image_o\ (f,\ \gamma)$

$SupIm_o$-$\lhd_o$-thm
$\vdash \forall\ \gamma\ f \bullet\ SupIm_o\ (\gamma \lhd_o f,\ \gamma) = SupIm_o\ (f,\ \gamma)$

$SSupIm_o$-$\lhd_o$-thm
$\vdash \forall\ \gamma\ f \bullet\ SSupIm_o\ (\gamma \lhd_o f,\ \gamma) = SSupIm_o\ (f,\ \gamma)$

ord_rec_thm $\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet\ f\ x = af\ ((x,\ \$<_o) \langle \lhd\ f)\ x$

ord_rec_thm2 $\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet\ f\ x = af\ (x \lhd_o f)\ x$

ord_rec_thm3 $\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet\ f\ (CombI\ x) = af\ (x \lhd_o f)\ x$

$Image_o$-$recursion\_thm$
$\vdash \forall\ af \bullet \exists\ f \bullet \forall\ x \bullet\ f\ (CombI\ x) = af\ (Image_o\ (f,\ x))\ x$

$plus_o$-0_o $\vdash \forall\ \beta \bullet\ \beta +_o 0_o = \beta$

lt_o-$\subseteq$ $\vdash \forall\ \beta\ \gamma \bullet\ \gamma <_o \beta \Rightarrow X_o\ \gamma \subseteq X_o\ \beta$

lt_o-$\subset$ $\vdash \forall\ \beta\ \gamma \bullet\ \gamma <_o \beta \Rightarrow X_o\ \gamma \subset X_o\ \beta$

$\leqslant_o$-$\subseteq$ $\vdash \forall\ \beta\ \gamma \bullet\ \gamma \leqslant_o \beta \Rightarrow X_o\ \gamma \subseteq X_o\ \beta$

$\leqslant_{oc}$-$refl$ $\vdash \forall\ \beta \bullet\ \beta \leqslant_{oc} \beta$

lt_o-$\leqslant_{oc}$ $\vdash \forall\ \beta\ \gamma \bullet\ \gamma <_o \beta \Rightarrow \gamma \leqslant_{oc} \beta$

$\leqslant_{oc}$-$trans$ $\vdash \forall\ \beta\ \gamma\ \eta \bullet\ \beta \leqslant_{oc} \gamma \wedge \gamma \leqslant_{oc} \eta \Rightarrow \beta \leqslant_{oc} \eta$

$\leqslant_{oc}$-$cases$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta \leqslant_{oc} \gamma \vee \gamma \leqslant_{oc} \beta$

lt_{oc}-$irrefl$
$\vdash \forall\ \beta \bullet\ \neg\ \beta <_{oc} \beta$

$\leqslant_{oc}$-$\neg$-lt_{oc}
$\vdash \forall\ \beta\ \gamma \bullet\ \beta \leqslant_{oc} \gamma \Rightarrow \neg\ \gamma <_{oc} \beta$

$\leqslant_{oc}$-$cases2$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta \leqslant_{oc} \gamma \Leftrightarrow \beta <_{oc} \gamma \vee \beta \sim_{oc} \gamma$

eq_{oc}-$refl$ $\vdash \forall\ \beta \bullet\ \beta \sim_{oc} \beta$

eq_{oc}-sym $\vdash \forall\ \beta\ \gamma \bullet\ \beta \sim_{oc} \gamma \Rightarrow \gamma \sim_{oc} \beta$

eq_{oc}-$trans$ $\vdash \forall\ \beta\ \gamma\ \eta \bullet\ \beta \sim_{oc} \gamma \wedge \gamma \sim_{oc} \eta \Rightarrow \beta \sim_{oc} \eta$

lt_{oc}-$trich$ $\vdash \forall\ \beta\ \gamma \bullet\ \beta <_{oc} \gamma \vee \gamma <_{oc} \beta \vee \beta \sim_{oc} \gamma$

$InitialOrdinal\_exists$
$\vdash \exists\ \beta \bullet\ InitialOrdinal\ \beta$

$InitialOrdinals\_exist$
$\vdash \forall\ \beta \bullet \exists\ \delta \bullet\ InitialOrdinal\ \delta \wedge \delta \sim_{oc} \beta$

$InitialOrdinal\_eq$
$\vdash \forall\ \beta\ \gamma$
$\bullet\ InitialOrdinal\ \beta \wedge InitialOrdinal\ \gamma \wedge \beta \sim_{oc} \gamma$
$\Rightarrow \beta = \gamma$

C INDEX