Inductive and Co-inductive Definitions in ProofPower

Roger Bishop Jones
rbj@rbjones.com

Abstract

Systematic facilities for a range of different kinds of inductive and co-inductive definitions of sets and types in ProofPower HOL.

Created 2004/07/15

Last Change Date: 2013/01/28 19:30:13

http://www.rbjones.com/rbjpub/pp/doc/t007.pdf

Id: t007.doc,v 1.31 2013/01/28 19:30:13 rbj Exp

Contents

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*. North Holland, 1977.

[2] R.D. Arthan. On Free Type Definitions in Z. In J.E. Nicholls, editor, *Z User Workshop, York 1991*. Springer-Verlag, 1992.

[3] T.E. Forster. *Set Theory with a Universal Set - Exploring an Untyped Universe*. Oxford University Press, 1992.

[4] Roger Bishop Jones. Illustrations of (Pseudo-)(Co-)Inductive Definitions. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t008.pdf`.

[5] Roger Bishop Jones. Inductive, Co-Inductive and Psuedo-(Co-)Inductive Definitions in ProofPower. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t007.pdf`.

[6] L. C. Paulson. A fixedpoint approach to (co)inductive and (co)datatype definitions. In C. Stirling G. Plotkin and M. Tofte, editors, *Proof, Language, and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

1 INTRODUCTION

Inductive definitions are so pervasive and diverse in logic, mathematics and computer science that it is doubtful that a comprehensive account could be given. In this document I attempt to provide support for some kinds of inductive type definition commonly used in computing at the same time as facilitating various other kinds of inductive definition of sets which are relevant to my interests in set theory and foundational studies. Some interest for me and perhaps some novelty for the reader may arise from the ecclectic mix of applications through which I have attempted to find common threads.

I began with a proof of the "Knaster-Tarski" theorem asserting that monotone functions over some power set have greatest and least fixed points, and this does provide at least one point common to all the material.

1.1 Pre-Topologies, Closures and Interiors

I'm groping here for the best language for describing what's going on.

It is natural to consider inductive definitions as arising from closures, and closure as a topological concept, though in fact the notion of closure is strictly broader than its use in topology. I have adopted apparently topological terminology where that seems appropriate, primarily by taking "open" as a dual for "closed" even in our present non-topological context. The kind of closure of interest here is strictly pre-topological, in the sense that these structures belong to a kind which is broader than the notion of topology (let's for the while call this a "pre-topology"), and all the interesting cases from our present perspective are pre-topologies which are not actually topologies.

To make this explanation more informative let us chose a definition of pre-topology. It won't do to generalise the idea of a topology as a set of open sets, because a pre-topology is not determined by its set of open sets. What we do instead is generalise the notion of a limit to that of an operator, or the notion of a limit point to that of a point of closure.

Think of a topology as being determined by a (monotonic) function which maps each subset of some universe into its closure, i.e. the smallest enclosing set which contains all its limit points. Such a map tells you what the open sets are (they are the complements of fixed points of the function), but not every such map determines a topology. It will fail to do so if, as in all the cases of present interest, the empty set is not one of its fixed points.

This gives a generalised notion of open and closed, which is perhaps best spoken of by using "closure point" as a generalisation of "limit point". A set is open if all its members are a among its closure points, and closed if it contains all its closure points.

In the pre-topologies of interest to us the empty set is never closed nor the universal set open. There are really only two sets in a pre-topology which are of interest from our present point of view, the closure of the empty set, which is the set defined inductively by the pre-topology, and the interior of the universal set, which is the set defined co-inductively by the pre-topology.

Having thus explained our talk of closures and interiors, we will have no further use for the notion of a pre-topology, and will work only with sets of operators, which may be thought of as defining pre-topologies in relation to which talk of closures and interiors may be understood.

1.2 Related Work

For a general introduction to inductive definitions see Aczel [1].

This work is conducted in a proof tool for higher order logic after the manner of LCF, and the literature concerning inductive definitions in similar contexts is therefore relevant. Several papers by Larry Paulson fall into this category, for example [6], in which may be found a more comprehensive guide to related work. I am aware of having taken from Paulson the idea of using the Knaster-Tarski fixedpoint theorem, but it is probable that his contribution, direct or indirect, is pervasive.

The particular system I have used is ProofPower, in relation to which the problem of "Free Types" in the Z specification language is relevant, and is discussed by Rob Arthan in [2]. So far however, the present work does not support any facilities specific to Z.

1.3 Intended Applications

Three kinds of application are approached:

1. support for defining (mutual) recursive datatypes where the character of the constructors is known but no particular types are known over which the constructions operate.
2. similar features for use in a context like set theory (in higher order logic) where it is essential to code up the constructors in that context (for example as required to prove Tarski's result about the definability of arithmetic or set theoretic truth in the languages of arithmetic or set theory).
3. support for a generalisation of what set theorists call "Hereditarily P" sets, i.e. sets which have property P all of whose members have property "Hereditarily P". The generalisation is to allow the constituents which must have the property to be defined other than by plain membership, as for example, in the "hereditarily pure functions", even though the members of such a function are not themselves functions (because of the way functions are usually coded up in set theory). In this case the relevant constutuents are the members of the domain and range of the function.

1.4 Formalities

Create new theory "fixp".

SML

```
open_theory "rbjmisc";
force_new_theory "fixp";
new_parent "U_orders";
new_parent "wf_relp";
(* new_parent "membership"; *)
force_new_pc "'fixp";
merge_pcs ["'savedthm_cs_∃_proof"] "'fixp";
set_merge_pcs ["rbjmisc", "'fixp"];
```

2 ORDERS AND BOUNDS

We need to talk about greatest lower bounds etc. This would have gone in "ordered_sets" but it doesn't follow the general treatment their, so for the time being its here.

First lets say what a lower bound is:

HOL Constant

$$\boldsymbol{IsLb} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a \to BOOL$$

$$\forall\ r\ s\ e\bullet\ IsLb\ r\ s\ e = \forall x\bullet\ x \in s \Rightarrow r\ e\ x$$

HOL Constant

$$\boldsymbol{IsUb} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a \to BOOL$$

$$\forall\ r\ s\ e\bullet\ IsUb\ r\ s\ e = \forall x\bullet\ x \in s \Rightarrow r\ x\ e$$

Often the domain of our relations does not have a unique maximal element, and so there may be no greatest lower bound for the empty set. However, this affects the applications rather than the concept:

HOL Constant

$$\boldsymbol{IsGlb} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a \to BOOL$$

$$\forall\ r\ s\ e\bullet\ IsGlb\ r\ s\ e \Leftrightarrow IsLb\ r\ s\ e \wedge \forall x\bullet\ IsLb\ r\ s\ x \Rightarrow r\ x\ e$$

HOL Constant

$$\boldsymbol{IsLub} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a \to BOOL$$

$$\forall\ r\ s\ e\bullet\ IsLub\ r\ s\ e \Leftrightarrow IsUb\ r\ s\ e \wedge \forall x\bullet\ IsUb\ r\ s\ x \Rightarrow r\ e\ x$$

HOL Constant

$$\boldsymbol{Lub} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a$$

$$\forall\ r\ s\bullet\ Lub\ r\ s = \epsilon x\bullet\ IsLub\ r\ s\ x$$

HOL Constant

$$\boldsymbol{Glb} : ('a \to 'a \to BOOL) \to 'a\ SET \to 'a$$

$$\forall\ r\ s\bullet\ Glb\ r\ s = \epsilon x\bullet\ IsGlb\ r\ s\ x$$

$$\boldsymbol{lub\_unique\_lemma} =$$
$$\vdash \forall\ X\ r\ x\ y\bullet\ Antisym\ (Universe, r) \Rightarrow IsLub\ r\ Y\ x \wedge IsLub\ r\ Y\ y \Rightarrow x = y$$

$$\boldsymbol{lub\_unique\_lemma2} =$$
$$\vdash \forall\ X\ r\ x\ y$$
$$\bullet\ Antisym\ (X, r) \wedge x \in X \wedge y \in X \Rightarrow IsLub\ r\ X\ x \wedge IsLub\ r\ X\ y \Rightarrow x = y$$

$$\boldsymbol{glb\_unique\_lemma} =$$
$$\vdash \forall\ X\ r\ x\ y\bullet\ Antisym\ (Universe, r) \Rightarrow IsGlb\ r\ X\ x \wedge IsGlb\ r\ X\ y \Rightarrow x = y$$

$$\boldsymbol{glb\_unique\_lemma2} =$$
$$\vdash \forall\ X\ r\ x\ y$$
$$\bullet\ Antisym\ (X, r) \wedge x \in X \wedge y \in X \Rightarrow IsGlb\ r\ X\ x \wedge IsGlb\ r\ X\ y \Rightarrow x = y$$

$$\textbf{\textit{isub\_sub\_lemma}} =$$
$$\vdash \forall\ X\ Y\ r\ y\bullet\ X \subseteq Y \wedge \mathit{IsUb}\ r\ Y\ y \Rightarrow \mathit{IsUb}\ r\ X\ y$$

$$\textbf{\textit{islb\_sub\_lemma}} =$$
$$\vdash \forall\ X\ Y\ r\ y\bullet\ X \subseteq Y \wedge \mathit{IsLb}\ r\ Y\ y \Rightarrow \mathit{IsLb}\ r\ X\ y$$

$$\textbf{\textit{isub\_trans\_lemma}} =$$
$$\vdash \forall\ X\ r\ x\ y\bullet\ \mathit{Trans}\ (\mathit{Universe},\ r) \wedge \mathit{IsUb}\ r\ X\ x \wedge r\ x\ y \Rightarrow \mathit{IsUb}\ r\ X\ y$$

$$\textbf{\textit{islb\_trans\_lemma}} =$$
$$\vdash \forall\ X\ r\ x\ y\bullet\ \mathit{Trans}\ (\mathit{Universe},\ r) \wedge \mathit{IsLb}\ r\ X\ y \wedge r\ x\ y \Rightarrow \mathit{IsLb}\ r\ X\ x$$

$$\textbf{\textit{islub\_sub\_lemma}} =$$
$$\vdash \forall\ X\ Y\ r\ x\ y\bullet\ X \subseteq Y \wedge \mathit{IsLub}\ r\ X\ x \wedge \mathit{IsLub}\ r\ Y\ y \Rightarrow r\ x\ y$$

$$\textbf{\textit{isglb\_sub\_lemma}} =$$
$$\vdash \forall\ X\ Y\ r\ x\ y\bullet\ X \subseteq Y \wedge \mathit{IsGlb}\ r\ X\ x \wedge \mathit{IsGlb}\ r\ Y\ y \Rightarrow r\ y\ x$$

$$\textbf{\textit{lub\_lub\_lemma}} =$$
$$\vdash \forall X\ r\bullet\ (\exists z\bullet\ \mathit{IsLub}\ r\ X\ z) \Rightarrow \mathit{IsLub}\ r\ X\ (\mathit{Lub}\ r\ X)$$

$$\textbf{\textit{glb\_glb\_lemma}} =$$
$$\vdash \forall X\ r\bullet\ (\exists z\bullet\ \mathit{IsGlb}\ r\ X\ z) \Rightarrow \mathit{IsGlb}\ r\ X\ (\mathit{Glb}\ r\ X)$$

$$\textbf{\textit{isglb\_glb\_lemma}} =$$
$$\vdash \forall\ X\ r\ z\bullet\ \mathit{Antisym}\ (\mathit{Universe},\ r) \Rightarrow \mathit{IsGlb}\ r\ X\ z \Rightarrow \mathit{Glb}\ r\ X = z$$

$$\textbf{\textit{isglb\_lub\_lemma}} =$$
$$\vdash \forall\ X\ r\ z\bullet\ \mathit{Antisym}\ (\mathit{Universe},\ r) \Rightarrow \mathit{IsLub}\ r\ X\ z \Rightarrow \mathit{Lub}\ r\ X = z$$

$$\textbf{\textit{ub\_ub\_lemma1}} =$$
$$\vdash \forall\ r\ X\ y\bullet\ \mathit{IsUb}\ r\ X\ y \Rightarrow (\forall\ x\bullet\ x \in X \Rightarrow r\ x\ y)$$

$$\textbf{\textit{lub\_ub\_lemma1}} =$$
$$\vdash \forall\ r\ X\ y\bullet\ \mathit{IsLub}\ r\ X\ y \Rightarrow (\forall\ x\bullet\ x \in X \Rightarrow r\ x\ y)$$

$$\textbf{\textit{lb\_lb\_lemma1}} =$$
$$\vdash \forall\ r\ X\ y\bullet\ \mathit{IsLb}\ r\ X\ y \Rightarrow (\forall\ x\bullet\ x \in X \Rightarrow r\ y\ x)$$

$$\textbf{\textit{glb\_lb\_lemma1}} =$$
$$\vdash \forall\ r\ X\ y\bullet\ \mathit{IsGlb}\ r\ X\ y \Rightarrow (\forall\ x\bullet\ x \in X \Rightarrow r\ y\ x)$$

$$\textbf{\textit{lin\_refl\_lub\_lemma}} =$$
$$\vdash \forall\ r\ X\ s\ l$$
$$\bullet\ \mathit{LinearOrder}\ (X,\ r) \wedge \mathit{Refl}\ (X,\ r) \wedge s \subseteq X \wedge l \in X \wedge \mathit{IsLub}\ r\ s\ l$$
$$\Rightarrow (\forall\ y\bullet\ y \in X \wedge r\ y\ l \wedge \neg\ r\ l\ y \Rightarrow (\exists\ z\bullet\ z \in s \wedge r\ y\ z))$$

HOL Constant

$\boldsymbol{NeGlbsExist} : ('a \to 'a \to BOOL) \to BOOL$

$\forall\ r\bullet\ NeGlbsExist\ r \Leftrightarrow \forall s\bullet\ (\exists d\bullet\ d \in s) \Rightarrow \exists e\bullet\ IsGlb\ r\ s\ e$

HOL Constant

$\boldsymbol{GlbsExist} : ('a \to 'a \to BOOL) \to BOOL$

$\forall\ r\bullet\ GlbsExist\ r \Leftrightarrow \forall s\bullet\ \exists e\bullet\ IsGlb\ r\ s\ e$

HOL Constant

$\boldsymbol{LubsExist} : ('a \to 'a \to BOOL) \to BOOL$

$\forall\ r\bullet\ LubsExist\ r \Leftrightarrow \forall s\bullet\ \exists e\bullet\ IsLub\ r\ s\ e$

$\boldsymbol{lub\_lub\_lemma2} =$

$\vdash \forall\ r\ X\bullet\ LubsExist\ r \Rightarrow IsLub\ r\ X\ (Lub\ r\ X)$

$\boldsymbol{glb\_glb\_lemma2} =$

$\vdash \forall\ r\ X\bullet\ GlbsExist\ r \Rightarrow IsGlb\ r\ X\ (Glb\ r\ X)$

$\boldsymbol{less\_lub\_lemma} =$

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow (\forall\ x\ X\bullet\ x \in X \Rightarrow r\ x\ (Lub\ r\ X))$

$\boldsymbol{gt\_glb\_lemma} =$

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow (\forall\ x\ X\bullet\ x \in X \Rightarrow r\ (Glb\ r\ X)\ x)$

$\boldsymbol{lub\_sub\_lemma} =$

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow (\forall\ X\ Y\bullet\ X \subseteq Y \Rightarrow r\ (Lub\ r\ X)\ (Lub\ r\ Y))$

$\boldsymbol{glb\_sub\_lemma} =$

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow (\forall\ X\ Y\bullet\ X \subseteq Y \Rightarrow r\ (Glb\ r\ Y)\ (Glb\ r\ X))$

$\boldsymbol{le\_islub\_lub\_lemma} =$

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow (\forall\ X\bullet\ IsLub\ r\ X\ (Lub\ r\ X))$

$\boldsymbol{ge\_isglb\_glb\_lemma} =$

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow (\forall\ X\bullet\ IsGlb\ r\ X\ (Glb\ r\ X))$

$\boldsymbol{lea\_islub\_lub\_lemma} =$

$\vdash \forall\ r$

$\bullet\ LubsExist\ r \wedge Antisym\ (Universe,\ r)$

$\Rightarrow (\forall\ X\ x\bullet\ IsLub\ r\ X\ x = Lub\ r\ X = x)$

$\boldsymbol{gea\_isglb\_glb\_lemma} =$

$\vdash \forall\ r$

$\bullet\ GlbsExist\ r \wedge Antisym\ (Universe,\ r)$

$\Rightarrow (\forall\ X\ x\bullet\ IsGlb\ r\ X\ x = Glb\ r\ X = x)$

3 FIXED POINTS

The general scheme involves finding closures of sets under varieties of operators. The operators take members or sets of members of elements of the set in question, and construct new values from them. Taking the closure of such a set under certain operations involves adding the things which can be constructed from elements in the set to the set untill there are no longer any new values which can be constructed.

In seeking to provide general support for such closure operations in a typed framework the diversity of the possible operations is an issue. For this reason, in addressing the fundamental problem for finding the required closure it is supposed that the operations over elements in the sets which are of interest have been compounded into a monotonic operator operating on a set and yeilding that same set augmented by the elements which can be constructed from them.

In the present type-theoretic context such an operator is always monotonic and unbounded, and therefore by the Knaster-Tarski fixedpoint theorem will have at least one fixed point.

In this section we prove the fixedpoint theorem and spin out some elementary consequences.

For reasons connected with the motivation of this work, the emphasis is on taking closures, and the concept of closure is made prominent. There is a dual system of terminology, at least insofar the dual of induction is called "co-induction" and though I am not strongly motivated by the aplications of the dual system, there is so little extra work in providing both that I have systematically done so, in the hope that applications may become apparent. In doing so I have used a "co" prefix whenever I know of no other terminology for the dual of a concept.

3.1 Monotonicity, Closure and Interior

Definition of the notion of a function over a powerset monotonic with respect to set inclusion. The Knaster-Tarski fixed point theorem applies to bounded monotonic functions, when considering only HOL "SET"s, as we do here, the boundedness condition is automatically satisfied.

HOL Constant

$$\boldsymbol{Monotonic} : ('a\ SET \to 'b\ SET) \to BOOL$$

$$\forall f \bullet\ Monotonic\ f \Leftrightarrow \forall x\ y \bullet\ x \subseteq y \Rightarrow f(x) \subseteq f(y)$$

SML

```
declare_infix (300, "ClosedUnder");
declare_infix (300, "OpenUnder");
```

HOL Constant

$$\$\boldsymbol{ClosedUnder} : 'a\ SET \to ('a\ SET \to 'a\ SET) \to BOOL$$

$$\forall f\ X \bullet\ X\ ClosedUnder\ f \Leftrightarrow f(X) \subseteq X$$

HOL Constant

$$\$\boldsymbol{OpenUnder} : 'a\ SET \to ('a\ SET \to 'a\ SET) \to BOOL$$

$$\forall f\ X \bullet\ X\ OpenUnder\ f \Leftrightarrow X \subseteq f(X)$$

It would be possible here to provide an induction principle based on the well-founded ordering by rank, in which the rank is the number of iteration of f starting with the empty set necessary to reach the element in question. However, I'm not convinved that this would be useful. Induction principles derived from more specific features of particular constructions are likely to be more useful in practice. I shall therefore hold off attemping such principles unless I find a need for them.

3.2 Least and Greatest Fixed Points

Now we define the conditions for being a least fixed point.

HOL Constant

$$\boldsymbol{IsLfp} : ('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a \to BOOL$$

$$\forall\ r\ f\ e\bullet\ IsLfp\ r\ f\ e \Leftrightarrow$$
$$\quad let\ fp = \{x \mid f\ x = x\}$$
$$\quad in\ e \in fp \wedge IsGlb\ r\ fp\ e$$

HOL Constant

$$\boldsymbol{IsGfp} : ('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a \to BOOL$$

$$\forall\ r\ f\ e\bullet\ IsGfp\ r\ f\ e \Leftrightarrow$$
$$\quad let\ fp = \{x \mid f\ x = x\}$$
$$\quad in\ e \in fp \wedge IsLub\ r\ fp\ e$$

$$\boldsymbol{islfp\_lemma1} =$$
$$\quad \vdash \forall\ r\ f\ e\bullet\ IsLfp\ r\ f\ e \Rightarrow e = f\ e \wedge (\forall\ x\bullet\ x = f\ x \Rightarrow r\ e\ x)$$

$$\boldsymbol{isgfp\_lemma1} =$$
$$\quad \vdash \forall\ r\ f\ e\bullet\ IsGfp\ r\ f\ e \Rightarrow e = f\ e \wedge (\forall\ x\bullet\ x = f\ x \Rightarrow r\ x\ e)$$

$$\boldsymbol{islfp\_lemma2} =$$
$$\quad \vdash \forall\ r\ f\ e\bullet\ IsLfp\ r\ f\ e \Rightarrow f\ e = e \wedge (\forall\ x\bullet\ f\ x = x \Rightarrow r\ e\ x)$$

$$\boldsymbol{isgfp\_lemma2} =$$
$$\quad \vdash \forall\ r\ f\ e\bullet\ IsGfp\ r\ f\ e \Rightarrow f\ e = e \wedge (\forall\ x\bullet\ f\ x = x \Rightarrow r\ x\ e)$$

$$\boldsymbol{islfp\_unique\_lemma} =$$
$$\quad \vdash \forall\ r\ f\ e\ d\bullet\ Antisym\ (Universe, r) \wedge IsLfp\ r\ f\ e \wedge IsLfp\ r\ f\ d \Rightarrow e = d$$

$$\boldsymbol{isgfp\_unique\_lemma} =$$
$$\quad \vdash \forall\ r\ f\ e\ d\bullet\ Antisym\ (Universe, r) \wedge IsGfp\ r\ f\ e \wedge IsGfp\ r\ f\ d \Rightarrow e = d$$

HOL Constant

$$\boldsymbol{Lfp} : ('a\ SET \to 'a\ SET) \to 'a\ SET$$

$$\forall f\bullet\ Lfp\ f = \bigcap\{X \mid X\ ClosedUnder\ f\}$$

Lfp gives a fixed point:

$$\begin{array}{|l} \boldsymbol{lfp\_fixedpoint\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow h\ (\mathit{Lfp}\ h) = \mathit{Lfp}\ h \end{array}$$

Lfp gives the least fixed point:

$$\begin{array}{|l} \boldsymbol{Lfp\_lfp\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall g \bullet h\ g = g \Rightarrow (\mathit{Lfp}\ h) \subseteq g \end{array}$$

HOL Constant

$$\begin{array}{|l} \boldsymbol{Gfp} : (\mathit{'a\ SET} \rightarrow \mathit{'a\ SET}) \rightarrow \mathit{'a\ SET} \\ \hline \forall f \bullet \mathit{Gfp}\ f = \bigcup \{X \mid X\ \mathit{OpenUnder}\ f\} \end{array}$$

Gfp gives a fixed point:

$$\begin{array}{|l} \boldsymbol{gfp\_fixedpoint\_thm} \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow h\ (\mathit{Gfp}\ h) = \mathit{Gfp}\ h \end{array}$$

Gfp gives the greatest fixed point:

$$\begin{array}{|l} \boldsymbol{gfp\_gfp\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall g \bullet h\ g = g \Rightarrow g \subseteq (\mathit{Gfp}\ h) \end{array}$$

3.3 Closure and Interior

$$\begin{array}{|l} \boldsymbol{lfp\_closed\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow (\mathit{Lfp}\ h)\ \mathit{ClosedUnder}\ h \\ \\ \boldsymbol{lfp\_closed\_thm1} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ x \bullet x \in h\ (\mathit{Lfp}\ h) \Rightarrow x \in \mathit{Lfp}\ h) \\ \\ \boldsymbol{lfp\_open\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow (\mathit{Lfp}\ h)\ \mathit{OpenUnder}\ h \\ \\ \boldsymbol{lfp\_open\_thm1} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ x \bullet x \in \mathit{Lfp}\ h \Rightarrow x \in h\ (\mathit{Lfp}\ h)) \end{array}$$

$$\begin{array}{|l} \boldsymbol{gfp\_closed\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow (\mathit{Gfp}\ h)\ \mathit{ClosedUnder}\ h \\ \boldsymbol{gfp\_closed\_thm1} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ x \bullet x \in h\ (\mathit{Gfp}\ h) \Rightarrow x \in \mathit{Gfp}\ h) \\ \boldsymbol{gfp\_open\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow (\mathit{Gfp}\ h)\ \mathit{OpenUnder}\ h \\ \boldsymbol{gfp\_open\_thm1} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ x \bullet x \in \mathit{Gfp}\ h \Rightarrow x \in h\ (\mathit{Gfp}\ h)) \end{array}$$

3.4 Induction and Co-induction

To prove something about the members of an inductively defined set you prove that all the operations which were used to define the set preserve the required property. In our case, where the operator is defined over sets of elements and we think of ourselves as starting out with the empty set, there is no "base case" for the induction (alternatively think of the base case as the application of the operator to the empty set).

For some property represented as a set s and a set of constructors represented as a function h the induction principle states that if the property is closed under the function then all the elements of the set inductively defined by the function (e.g. its least fixed point) have the property.

The least fixed point induction principle may therefore be expressed:

$$\begin{array}{l}\boldsymbol{lfp\_induction\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall s \bullet\ s\ \mathit{ClosedUnder}\ h \Rightarrow (\mathit{Lfp}\ h) \subseteq s \\ \boldsymbol{lfp\_induction\_thm1} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall s \bullet\ h\ s \subseteq s \Rightarrow (\mathit{Lfp}\ h) \subseteq s\end{array}$$

This is the corresponding theorem for greatest fixed point to the "induction" principle for least fixed points.

$$\begin{array}{l}\boldsymbol{gfp\_coinduction\_thm} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall s \bullet\ s\ \mathit{OpenUnder}\ h \Rightarrow s \subseteq (\mathit{Gfp}\ h) \\ \boldsymbol{gfp\_coinduction\_thm1} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ s \bullet\ s \subseteq h\ s \Rightarrow s \subseteq \mathit{Gfp}\ h)\end{array}$$

These basic induction and co-induction theorems can be strengthened. The induction principle tells us that any collection having the relevant closure property contains the closure.

The following allows properties to be established which are not themselves closed.

$$\begin{array}{l}\boldsymbol{lfp\_induction\_thm2} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall s \bullet\ (s \cap (\mathit{Lfp}\ h))\ \mathit{ClosedUnder}\ h \Rightarrow (\mathit{Lfp}\ h) \subseteq s \\ \boldsymbol{gfp\_coinduction\_thm2} = \\ \quad \vdash \forall h \bullet \mathit{Monotonic}\ h \Rightarrow \forall s \bullet\ (s \cup (\mathit{Gfp}\ h))\ \mathit{OpenUnder}\ h \Rightarrow s \subseteq (\mathit{Gfp}\ h)\end{array}$$

Above one must show that the intersection of s and the least fixed point of h is closed under h.

It suffices however to show that the image of that set under h is contained in s:

$$\begin{array}{l}\boldsymbol{lfp\_induction\_thm3} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ s \bullet\ h\ (s \cap \mathit{Lfp}\ h) \subseteq s \Rightarrow \mathit{Lfp}\ h \subseteq s) \\ \boldsymbol{gfp\_coinduction\_thm3} = \\ \quad \vdash \forall\ h \bullet \mathit{Monotonic}\ h \Rightarrow (\forall\ s \bullet\ s \subseteq h\ (s \cup \mathit{Gfp}\ h) \Rightarrow s \subseteq \mathit{Gfp}\ h)\end{array}$$

3.5 Monotonicity Against Other Relations

The above considers only functions on sets which are monotonic relative to the subset relationship.

In order to get fixed points the relations relative to which our functions are monotone must have appropriate kinds of completeness. Ideally they will be simply "complete" i.e. having supremum

and infimum of arbitrary sets of elements. This makes the domain into a complete lattice, but since we focus on the relation and do not have operators for supremum and infimum in the structure we are considering, what we are doing doesn't look like lattice theory.

However, we sometimes want to obtain fixed points relative to orderings which are not complete. We begin with chain-complete partial orders. I should for consistency have done chain co-complete partial orders as well but that didn't occur to me at the time.

3.6 Reflexive Partial Orders

It is simpler to do this specifically for reflexive partial orders, whereas the theory of orders already available is not specific to reflexive orders. The notion of of reflexive partial order is therefore defined first, and various relevant concepts (upper and lower bounds etc.) are introduced in this context.

Since the orders considered here are reflexive we will use the symbol $\leq_v$ as a variable ranging over these relations. Note however that the theory *ordered_sets*, on which we draw, uses $<<$. Note also that we do not follow generally here the practice of reasoning about relations over sets. Usually we reason about relations over whole types.

In the theory *ordered_sets* ordering relations are treated in many areas with indifference as to whether they are reflexive or strict or inbetween. Here we work primarily with reflexive partial orders. The distinction is marked (perhaps not consistently) by the choice of symbol for relation variable. Where a reflexive relation is intended the symbol $\leq_v$ is used, where the relation need not be reflexive we follow *ordered_sets* in using $<<$. Where I have used r this is prior to my introducing this scheme and the relation is probably reflexive.

SML

```
declare_infix (300, "≤v");
```

HOL Constant

$$\boldsymbol{Rpo} : ('a\ SET) \times ('a \to 'a \to BOOL) \to BOOL$$

$$\forall r \bullet\ Rpo\ r \Leftrightarrow PartialOrder\ r \wedge Refl\ r$$

$$\boldsymbol{rpo\_fc\_clauses} =$$
$$\vdash \forall\ (X, \$\leq_v)$$
$$\bullet\ Rpo\ (X, \$\leq_v)$$
$$\Rightarrow (\forall\ x\ y \bullet\ x \in X \wedge y \in X \wedge x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$$
$$\wedge (\forall\ x\ y\ z \bullet\ x \in X \wedge y \in X \wedge z \in X \wedge x \leq_v y \wedge y \leq_v z \Rightarrow x \leq_v z)$$
$$\wedge (\forall\ x \bullet\ x \in X \Rightarrow x \leq_v x)$$

HOL Constant

$$\boldsymbol{RpoU}: ('a \to 'a \to BOOL) \to BOOL$$

$$\forall\ r \bullet\ RpoU\ r \Leftrightarrow Rpo\ (Universe, r)$$

rpou_fc_clauses =

$\vdash \forall\ \$\leq_v$

$\bullet$ $Rpo\ (Universe,\ \$\leq_v)$

$\Rightarrow (\forall\ x\ y\bullet\ x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$

$\wedge\ (\forall\ x\ y\ z\bullet\ x \leq_v y \wedge y \leq_v z \Rightarrow x \leq_v z)$

$\wedge\ (\forall\ x\bullet\ x \leq_v x)$

rpou_fc_clauses2 =

$\vdash \forall\ \$\leq_v$

$\bullet$ $RpoU\ \$\leq_v$

$\Rightarrow (\forall\ x\ y\bullet\ x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$

$\wedge\ (\forall\ x\ y\ z\bullet\ x \leq_v y \wedge y \leq_v z \Rightarrow x \leq_v z)$

$\wedge\ (\forall\ x\bullet\ x \leq_v x)$

rpo_antisym_lemma =

$\vdash \forall\ X\ r\ f\bullet\ Rpo\ (X,\ r) \Rightarrow Antisym\ (X,\ r)$

rpo_∩_lemma =

$\vdash \forall\ X\ Y\ r\bullet\ Rpo\ (X,\ r) \wedge Rpo\ (Y,\ r) \Rightarrow Rpo\ (X \cap Y,\ r)$

HOL Constant

Lfp$_c$: $('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a$

$\forall\ r\ f\bullet\ (\exists\ e\bullet\ IsLfp\ r\ f\ e) \Rightarrow IsLfp\ r\ f\ (Lfp_c\ r\ f)$

HOL Constant

Gfp$_c$: $('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a$

$\forall\ r\ f\bullet\ (\exists\ e\bullet\ IsGfp\ r\ f\ e) \Rightarrow IsGfp\ r\ f\ (Gfp_c\ r\ f)$

islfp_unique_lemma2 =

$\vdash \forall\ r\ f\ e\bullet\ Antisym\ (Universe,\ r) \wedge IsLfp\ r\ f\ e \Rightarrow Lfp_c\ r\ f = e$

isgfp_unique_lemma2 =

$\vdash \forall\ r\ f\ e\bullet\ Antisym\ (Universe,\ r) \wedge IsGfp\ r\ f\ e \Rightarrow Gfp_c\ r\ f = e$

3.6.1 Directed Sets

These definitions are not actually used.

We need the concept of a directed set.

HOL Constant

Directed : $('a \to 'a \to BOOL) \to 'a\ SET \to BOOL$

$\forall\ r\ s\bullet\ Directed\ r\ s \Leftrightarrow \forall x\ y\bullet\ x \in s \wedge y \in s \Rightarrow \exists z\bullet\ z \in s \wedge IsUb\ r\ \{x;\ y\}\ z$

Which we use to define the property of having directed upper bounds:

HOL Constant

$$\boldsymbol{DirectedUb} : ('a \to 'a \to BOOL) \to BOOL$$

$$\forall\ r \bullet\ DirectedUb\ r \Leftrightarrow \forall s \bullet\ Directed\ r\ s \Rightarrow \exists x \bullet\ IsUb\ r\ s\ x$$

3.7 Monotonicity

We need to establish the existence of least fixed points of monotone functions and so we want a result to the effect that the greatest lower bound of closed elements is a fixed point. A closed element is one which maps to a point below itself in the ordering.

HOL Constant

$$\boldsymbol{Increasing} : ('a \to 'a \to BOOL) \to ('b \to 'b \to BOOL) \to ('a \to 'b) \to BOOL$$

$$\forall\ r\ s\ f \bullet\ Increasing\ r\ s\ f \Leftrightarrow \forall x\ y \bullet\ r\ x\ y \Rightarrow s\ (f\ x)\ (f\ y)$$

$$\boldsymbol{increasing\_funcomp\_thm} =$$
$$\vdash \forall\ r1\ r2\ r3\ f\ g$$
$$\bullet\ Increasing\ r1\ r2\ f \wedge Increasing\ r2\ r3\ g \Rightarrow Increasing\ r1\ r3\ (g\ o\ f)$$

$$\boldsymbol{increasing\_funcomp\_thm2} =$$
$$\vdash \forall\ r1\ r2\ r3\ f\ g$$
$$\bullet\ Increasing\ r1\ r2\ f \wedge Increasing\ r2\ r3\ g \Rightarrow Increasing\ r1\ r3\ (\lambda x \bullet\ g(f(x)))$$

Since the least fixed point will be a greatest lower bound, we need to know that greatest lower bounds exist to prove that there is a fixed point. Since we don't have a top element we assert only the existence of a greatest lower bound for non-empty sets.

$$\boldsymbol{mono\_fixp\_thm} =$$
$$\vdash \forall\ f\ r$$
$$\bullet\ Refl\ (Universe,\ r)$$
$$\wedge\ (\forall\ x\ y \bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$$
$$\wedge\ trans\ r$$
$$\wedge\ Increasing\ r\ r\ f$$
$$\wedge\ NeGlbsExist\ r$$
$$\wedge\ (\exists\ x \bullet\ r\ (f\ x)\ x)$$
$$\Rightarrow (\exists\ e \bullet\ IsGlb\ r\ \{x | r\ (f\ x)\ x\}\ e \wedge IsLfp\ r\ f\ e)$$

$$\boldsymbol{mono\_fixp\_thm2} =$$
$$\vdash \forall\ f\ r$$
$$\bullet\ Refl\ (Universe,\ r)$$
$$\wedge\ (\forall\ x\ y \bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$$
$$\wedge\ trans\ r$$
$$\wedge\ Increasing\ r\ r\ f$$
$$\wedge\ GlbsExist\ r$$

$\Rightarrow (\exists\ e\bullet\ IsGlb\ r\ \{x|r\ (f\ x)\ x\}\ e \wedge IsLfp\ r\ f\ e)$

mono_fixp_thm3 =
$\vdash \forall\ f\ r$
$\bullet\ Refl\ (Universe,\ r)$
$\wedge (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$
$\wedge\ trans\ r$
$\wedge\ Increasing\ r\ r\ f$
$\wedge\ LubsExist\ r$
$\Rightarrow (\exists\ e\bullet\ IsLub\ r\ \{x|r\ x\ (f\ x)\}\ e \wedge IsGfp\ r\ f\ e)$

4 CHAIN COMPLETE PARTIAL ORDERS

A ccpo is a chain-complete partial order. The proof of the fixed point result in this context is more difficult than in the case of a complete lattice, and the following version of it is a kludge, but will have to do for now.

The main point of this section is to obtain a least fixed point theorem for monotone functions over chain complete partial orders (CCPOs).

HOL Constant
ChainComplete : $('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$\forall\ X\ r\bullet\ ChainComplete\ (X,\ r) \Leftrightarrow \forall Y\bullet\ Y \subseteq X \wedge LinearOrder\ (Y,\ r) \Rightarrow$
$\exists x\bullet\ x \in X \wedge IsLub\ r\ Y\ x$

cc_lub_lemma =
$\vdash \forall\ X\ r$
$\bullet\ ChainComplete\ (X,\ r)$
$\Rightarrow (\forall\ Y\bullet\ Y \subseteq X \wedge LinearOrder\ (Y,\ r) \Rightarrow IsLub\ r\ Y\ (Lub\ r\ Y))$

ccu_lub_lemma =
$\vdash \forall\ r$
$\bullet\ ChainComplete\ (Universe,\ r)$
$\Rightarrow (\forall\ Y\bullet\ LinearOrder\ (Y,\ r) \Rightarrow IsLub\ r\ Y\ (Lub\ r\ Y))$

HOL Constant
CcRpo : $('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$\forall\ r\bullet\ CcRpo\ r \Leftrightarrow Rpo\ r \wedge ChainComplete\ r$

HOL Constant
CcRpoU : $('a \to 'a \to BOOL) \to BOOL$

$\forall\ r\bullet\ CcRpoU\ r \Leftrightarrow CcRpo\ (Universe,\ r)$

ccrpou_fc_clauses =

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow ChainComplete\ (Universe,\ r) \wedge (\forall\ x \bullet\ r\ x\ x) \wedge (\forall\ x\ y\ z \bullet\ r\ x\ y \wedge r\ y\ z \Rightarrow r\ x\ z) \wedge (\forall\ x\ y \bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$$

ccrpou_lub_lemma =

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow (\forall\ Y \bullet\ LinearOrder\ (Y,\ r) \Rightarrow IsLub\ r\ Y\ (Lub\ r\ Y))$$

ccrpou_lub_lemma2 =

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow (\forall\ Y \bullet\ LinearOrder\ (Y,\ r) \Rightarrow (\forall\ x \bullet\ x \in Y \Rightarrow r\ x\ (Lub\ r\ Y)))$$

ccrpou_lub_unique_lemma =

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow (\forall\ Y \bullet\ LinearOrder\ (Y,\ r) \Rightarrow (\forall\ x \bullet\ IsLub\ r\ Y\ x \Rightarrow x = Lub\ r\ Y))$$

HOL Constant

CRpo : $('a \to 'a \to BOOL) \to BOOL$

$$\forall\ r \bullet\ CRpo\ r \Leftrightarrow Rpo\ (Universe,\ r) \wedge GlbsExist\ r$$

HOL Constant

FClosed : $('a \to 'a) \to 'a\ SET \to BOOL$

$$\forall\ f\ X \bullet\ FClosed\ f\ X \Leftrightarrow (\forall x \bullet\ x \in X \Rightarrow f\ x \in X)$$

fclosed_universe_lemma =

$$\vdash \forall\ f \bullet\ FClosed\ f\ Universe$$

fclosed_∩_lemma =

$$\vdash \forall\ X\ Y\ f \bullet\ FClosed\ f\ X \wedge FClosed\ f\ Y \Rightarrow FClosed\ f\ (X \cap Y)$$

HOL Constant

FChainClosed : $('a \to 'a) \to ('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$$\forall\ f\ X\ r \bullet\ FChainClosed\ f\ (X,\ r) \Leftrightarrow FClosed\ f\ X \wedge ChainComplete\ (X,\ r)$$

ccrpou_fchainclosed_lemma =

$\vdash \forall\ r\bullet\ CcRpoU\ r \Rightarrow FChainClosed\ f\ (Universe,\ r)$

ccrpou_fchainclosed_lemma2 =

$\vdash \forall\ r\ X$
$\bullet\ CcRpoU\ r \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow (\forall\ s\bullet\ s \subseteq X \wedge LinearOrder\ (s,\ r) \Rightarrow Lub\ r\ s \in X)$

HOL Constant

FChain : $('a \to 'a) \to ('a\ SET \times ('a \to 'a \to BOOL)) \to 'a\ SET$

$\forall\ f\ X\ r\bullet\ FChain\ f\ (X,\ r) = \bigcap \{Y \mid Y \subseteq X \wedge FChainClosed\ f\ (Y,\ r)\}$

HOL Constant

FChainU : $('a \to 'a) \to ('a \to 'a \to BOOL) \to 'a\ SET$

$\forall\ f\ r\bullet\ FChainU\ f\ r = FChain\ f\ (Universe,\ r)$

fcc_fchain_lemma1 =

$\vdash (\exists\ V\bullet\ V \in W) \wedge (\forall\ V\bullet\ V \in W \Rightarrow V \subseteq X) \wedge Y \subseteq \bigcap W \Rightarrow Y \subseteq X$

fchain_lemma1 =

$\vdash \forall\ X\ r\ f\bullet\ FChainClosed\ f\ (X,\ r) \Rightarrow FChain\ f\ (X,\ r) \subseteq X$

fchain_fchainclosed_lemma =

$\vdash \forall\ X\ r\ f$
$\bullet\ Antisym\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow FChainClosed\ f\ (FChain\ f\ (X,\ r),\ r)$

fchain_fchainclosed_lemma2 =

$\vdash \forall\ X\ r\ f$
$\bullet\ Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow FChainClosed\ f\ (FChain\ f\ (X,\ r),\ r)$

ccrpou_fcclosed_fc_lemma =

$\vdash \forall\ r\ f\bullet\ CcRpoU\ r \Rightarrow FChainClosed\ f\ (FChainU\ f\ r,\ r)$

ccrpou_fcclosed_fc_lemma2 =

$\vdash \forall\ r\ f\ X$
$\bullet\ CcRpoU\ r \wedge X \subseteq FChainU\ f\ r \wedge LinearOrder\ (X,\ r)$
$\Rightarrow Lub\ r\ X \in FChainU\ f\ r$

$$
\begin{array}{l}
\boldsymbol{fchain\_lemma2} = \\
\quad \vdash \forall\ X\ Y\ r\ f \\
\quad\quad \bullet\ Rpo\ (Universe,\ r) \\
\quad\quad\quad\quad \wedge\ FChainClosed\ f\ (X,\ r) \\
\quad\quad\quad\quad \wedge\ Y \subseteq FChain\ f\ (X,\ r) \\
\quad\quad\quad\quad \wedge\ FChainClosed\ f\ (Y,\ r) \\
\quad\quad\quad\ \Rightarrow\ Y = FChain\ f\ (X,\ r)
\end{array}
$$

The last lemma is in effect an induction principle which is used several times in the proof which follows of the existence of least fixed points for monotone functors over chain-complete orders. Though I don't have a plan to rationalise that proof, I propose here to improve support for induction over $FChain$s.

First I recast the above lemma to make it look more explicitly like an induction principle. I am interested in the use of induction to prove results about least fixed points of functions which are monotone relative to some ordering which is complete over the whole type (though chain completeness suffices). The inductive part of such a proof will establish a property common to the elements of the $FChain$ generated by the function starting from the bottom element. In principle we could have an induction principle which yields a property of the least fixed point, but this would be less useful in general than proving the property of the whole $FChain$. This will allow proofs to be broken into smaller pieces than would otherwise be the case.

First $fchain\_lemma2$ is simplified for the case that the relation is chain complete over the type.

$$
\begin{array}{l}
\boldsymbol{fchain\_lemma3} = \\
\quad \vdash \forall\ X\ Y\ r\ f \\
\quad\quad \bullet\ CcRpoU\ r \wedge Y \subseteq FChainU\ f\ r \wedge FChainClosed\ f\ (Y,\ r) \Rightarrow Y = FChainU\ f\ r
\end{array}
$$

Now we make this look more like an induction principle. First, retaining the set theoretic casting, thinking of Y as the extension of some property which is to be proven of all elements in the $FChain$ (in this case Y need not be a subset of the chain):

$$
\begin{array}{l}
\boldsymbol{fchain\_lemma4} = \\
\quad \vdash \forall\ r\ f\ Y \\
\quad\quad \bullet\ CcRpoU\ r \\
\quad\quad\quad\quad \wedge\ FClosed\ f\ (Y \cap FChainU\ f\ r) \\
\quad\quad\quad\quad \wedge\ ChainComplete\ (Y \cap FChainU\ f\ r,\ r) \\
\quad\quad\quad\ \Rightarrow FChainU\ f\ r \subseteq Y
\end{array}
$$

Now we come to a formulation suitable for use in a typical induction tactic, which expects the property to be obtained by abstraction on the conclusion of the current goal.

$$
\begin{array}{l}
\boldsymbol{fchainu\_lemma1} = \\
\quad \vdash \forall\ f\ r\ x\bullet\ x \in FChainU\ f\ r \Rightarrow f\ x \in FChainU\ f\ r \\
\\
\boldsymbol{fchainu\_induction\_thm1} = \\
\quad \vdash \forall\ r\ f\ p \\
\quad\quad \bullet\ CcRpoU\ r \\
\quad\quad\quad\quad \wedge\ (\forall\ x\bullet\ x \in FChainU\ f\ r \wedge p\ x \Rightarrow p\ (f\ x)) \\
\quad\quad\quad\quad \wedge\ (\forall\ s\bullet\ s \subseteq FChainU\ f\ r \wedge (\forall\ y\bullet\ y \in s \Rightarrow p\ y) \Rightarrow p\ (Lub\ r\ s))
\end{array}
$$

$\Rightarrow x \in FChainU\ f\ r$
$\Rightarrow p\ x$

This induction theorem requires a proof that the property is chain-complete over the relevant $FChain$. However, in this context a "course of values" induction theorem should be realisable, in which the hypothesis of the step case is stronger. This amounts to the observation that the relevant r is well-founded on the $FChain$. The statement and proof of the course of values induction principle is deferred a while because it requires a result not yet obtained.

The following lemmas leading towards the fixed point theorem are proven by the method supported by $fchain\_lemma2$

$ccrpo\_fixp\_lemma1$ =
$\vdash \forall\ X\ r\ f$
$\bullet\ Increasing\ r\ r\ f$
$\Rightarrow FClosed\ f\ \{x | x \in FChain\ f\ (X,\ r) \wedge (r\ x\ (f\ x) \vee x = f\ x)\}$

$ccrpo\_fixp\_lemma2$ =
$\vdash \forall\ X\ r\ f$
$\bullet\ Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow ChainComplete\ (\{z | z \in FChain\ f\ (X,\ r) \wedge (r\ z\ (f\ z) \vee z = f\ z)\},\ r)$

$ccrpo\_fixp\_lemma3$ =
$\vdash \forall\ X\ r\ f$
$\bullet\ Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow FChainClosed$
f
$(\{z | z \in FChain\ f\ (X,\ r) \wedge (r\ z\ (f\ z) \vee z = f\ z)\},\ r)$

$ccrpo\_fixp\_lemma4$ =
$\vdash \forall\ X\ r\ f$
$\bullet\ Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r)$
$\Rightarrow (\forall\ z\bullet\ z \in FChain\ f\ (X,\ r) \Rightarrow r\ z\ (f\ z) \vee z = f\ z)$

This result is needed to establish course of values induction over $FChain$s, for which purpose the following variant is handy.

$ccrpou\_fchainu\_lemma1$ =
$\vdash \forall\ r\ f\bullet\ CcRpoU\ r \wedge Increasing\ r\ r\ f$
$\Rightarrow (\forall\ z\bullet\ z \in FChainU\ f\ r \Rightarrow r\ z\ (f\ z))$

HOL Constant
$Extreme$: $((\prime a \to \prime a) \times (\prime a\ SET \times (\prime a \to \prime a \to BOOL))) \to \prime a\ SET$

$\forall\ f\ X\ \$\leq_v\bullet\ Extreme\ (f,\ X,\ \$\leq_v) =$
$\{x \mid x \in FChain\ f\ (X,\ \$\leq_v) \wedge \forall y\bullet\ y \in FChain\ f\ (X,\ \$\leq_v)$
$\Rightarrow y \leq_v x \wedge \neg\ y = x \Rightarrow f\ y \leq_v x\}$

HOL Constant

$$\boldsymbol{S} : (('a \to 'a) \times ('a\ SET \times ('a \to 'a \to BOOL))) \to 'a \to 'a\ SET$$

$$\forall\ f\ X\ \$\leq_v\ x\bullet\ S\ (f,\ X,\ \$\leq_v)\ x =$$
$$\{y \mid y \in FChain\ f\ (X,\ \$\leq_v) \wedge (y \leq_v x \vee f\ x \leq_v y)\}$$

ccrpo_fixp_lemma5 =
$$\vdash \forall\ X\ \$\leq_v\ f\ x$$
$$\bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f$$
$$\wedge\ Rpo\ (Universe,\ \$\leq_v)$$
$$\wedge\ FChainClosed\ f\ (X,\ \$\leq_v)$$
$$\wedge\ x \in Extreme\ (f,\ X,\ \$\leq_v)$$
$$\Rightarrow\ FClosed\ f\ (S\ (f,\ X,\ \$\leq_v)\ x)$$

ccrpo_fixp_lemma6 =
$$\vdash \forall\ X\ \$\leq_v\ f\ x$$
$$\bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f$$
$$\wedge\ Rpo\ (Universe,\ \$\leq_v)$$
$$\wedge\ FChainClosed\ f\ (X,\ \$\leq_v)$$
$$\wedge\ x \in Extreme\ (f,\ X,\ \$\leq_v)$$
$$\Rightarrow\ ChainComplete\ (S\ (f,\ X,\ \$\leq_v)\ x,\ \$\leq_v)$$

ccrpo_fixp_lemma7 =
$$\vdash \forall\ X\ \$\leq_v\ f\ x$$
$$\bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f$$
$$\wedge\ Rpo\ (Universe,\ \$\leq_v)$$
$$\wedge\ FChainClosed\ f\ (X,\ \$\leq_v)$$
$$\wedge\ x \in Extreme\ (f,\ X,\ \$\leq_v)$$
$$\Rightarrow\ FChainClosed\ f\ (S\ (f,\ X,\ \$\leq_v)\ x,\ \$\leq_v)$$

ccrpo_fixp_lemma8 =
$$\vdash \forall\ X\ \$\leq_v\ f$$
$$\bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f$$
$$\wedge\ Rpo\ (Universe,\ \$\leq_v)$$
$$\wedge\ FChainClosed\ f\ (X,\ \$\leq_v)$$
$$\wedge\ x \in Extreme\ (f,\ X,\ \$\leq_v)$$
$$\Rightarrow\ FChain\ f\ (X,\ \$\leq_v) = S\ (f,\ X,\ \$\leq_v)\ x$$

ccrpo_fixp_lemma9 =
$$\vdash \forall\ X\ \$\leq_v\ f$$
$$\bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f$$
$$\wedge\ Rpo\ (Universe,\ \$\leq_v)$$
$$\wedge\ FChainClosed\ f\ (X,\ \$\leq_v)$$
$$\Rightarrow\ FClosed\ f\ (Extreme\ (f,\ X,\ \$\leq_v))$$

$$
\begin{array}{l}
\\
\textbf{\textit{ccrpo\_fixp\_lemma10}} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\qquad \bullet\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \wedge\ \mathit{Rpo}\ (\mathit{Universe},\ \$\leq_v) \\
\qquad\qquad \wedge\ \mathit{FChainClosed}\ f\ (X,\ \$\leq_v) \\
\qquad\qquad \Rightarrow\ \mathit{ChainComplete}\ (\mathit{Extreme}\ (f,\ X,\ \$\leq_v),\ \$\leq_v) \\
\\
\textbf{\textit{ccrpo\_fixp\_lemma11}} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f\ x \\
\qquad \bullet\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \wedge\ \mathit{Rpo}\ (\mathit{Universe},\ \$\leq_v) \\
\qquad\qquad \wedge\ \mathit{FChainClosed}\ f\ (X,\ \$\leq_v) \\
\qquad\qquad \Rightarrow\ \mathit{FChainClosed}\ f\ (\mathit{Extreme}\ (f,\ X,\ \$\leq_v),\ \$\leq_v) \\
\\
\textbf{\textit{ccrpo\_fixp\_lemma12}} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\qquad \bullet\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \wedge\ \mathit{Rpo}\ (\mathit{Universe},\ \$\leq_v) \\
\qquad\qquad \wedge\ \mathit{FChainClosed}\ f\ (X,\ \$\leq_v) \\
\qquad\qquad \Rightarrow\ \mathit{FChain}\ f\ (X,\ \$\leq_v) = \mathit{Extreme}\ (f,\ X,\ \$\leq_v)
\end{array}
$$

The above result may be rendered as a property of $FChainU$s (which we need to prove the "course of values" induction principle).

$$
\begin{array}{l}
\textbf{\textit{ccrpou\_fchainu\_lemma2}} = \\
\quad \vdash \forall\ \$\leq_v\ f \\
\qquad \bullet\ \mathit{CcRpoU}\ \$\leq_v \wedge\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \Rightarrow (\forall\ x\ y \\
\qquad\qquad \bullet\ x \in \mathit{FChainU}\ f\ \$\leq_v \wedge\ y \in \mathit{FChainU}\ f\ \$\leq_v \\
\qquad\qquad\qquad \Rightarrow\ y \leq_v\ x \wedge \neg\ y = x \\
\qquad\qquad\qquad \Rightarrow\ f\ y \leq_v\ x) \\
\\
\textbf{\textit{ccrpou\_fchainu\_lemma2b}} = \\
\quad \vdash \forall\ \$\leq_v\ f \\
\qquad \bullet\ \mathit{CcRpoU}\ \$\leq_v \wedge\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \Rightarrow (\forall\ x\ y \\
\qquad\qquad \bullet\ x \in \mathit{FChainU}\ f\ \$\leq_v \wedge\ y \in \mathit{FChainU}\ f\ \$\leq_v \\
\qquad\qquad\qquad \Rightarrow\ y \leq_v\ x \wedge \neg\ x \leq_v\ y \\
\qquad\qquad\qquad \Rightarrow\ f\ y \leq_v\ x) \\
\\
\textbf{\textit{ccrpo\_fixp\_lemma13}} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\qquad \bullet\ \mathit{Increasing}\ \$\leq_v\ \$\leq_v\ f \\
\qquad\qquad \wedge\ \mathit{Rpo}\ (\mathit{Universe},\ \$\leq_v) \\
\qquad\qquad \wedge\ \mathit{FChainClosed}\ f\ (X,\ \$\leq_v)
\end{array}
$$

$\Rightarrow$ *FChain* f $(X, \$\leq_v)$
$= \{x$
$|x \in$ *Extreme* $(f, X, \$\leq_v) \wedge$ *FChain* f $(X, \$\leq_v) \subseteq S$ $(f, X, \$\leq_v)$ $x\}$

ccrpo_fixp_lemma14 =
$\vdash \forall X\ \$\leq_v f$
• *Increasing* $\$\leq_v\ \$\leq_v f$
$\wedge$ *Rpo* $(Universe, \$\leq_v)$
$\wedge$ *FChainClosed* f $(X, \$\leq_v)$
$\Rightarrow (\forall x\ y$
• $\{x; y\} \subseteq$ *FChain* f $(X, \$\leq_v)$
$\Rightarrow (x \leq_v f\ x \vee x = f\ x)$
$\wedge (y \leq_v x \wedge \neg\ y = x \Rightarrow f\ y \leq_v x)$
$\wedge (y \leq_v x \vee f\ x \leq_v y))$

ccrpo_fixp_lemma15 =
$\vdash \forall X\ \$\leq_v f$
• *Increasing* $\$\leq_v\ \$\leq_v f$
$\wedge$ *Rpo* $(Universe, \$\leq_v)$
$\wedge$ *FChainClosed* f $(X, \$\leq_v)$
$\Rightarrow$ *Trich* $(FChain\ f\ (X, \$\leq_v), \$\leq_v)$

ccrpo_fixp_lemma16 =
$\vdash \forall X\ \$\leq_v f$
• *Increasing* $\$\leq_v\ \$\leq_v f$
$\wedge$ *Rpo* $(Universe, \$\leq_v)$
$\wedge$ *FChainClosed* f $(X, \$\leq_v)$
$\Rightarrow$ *LinearOrder* $(FChain\ f\ (X, \$\leq_v), \$\leq_v)$

ccrpou_fchainu_linear_lemma =
$\vdash \forall \$\leq_v f$
• *CcRpoU* $\$\leq_v \wedge$ *Increasing* $\$\leq_v\ \$\leq_v f$
$\Rightarrow$ *LinearOrder* $(FChainU\ f\ \$\leq_v, \$\leq_v)$

ccrpou_fchainu_linear_lemma2 =
$\vdash \forall X\ \$\leq_v f$
• *CcRpoU* $\$\leq_v \wedge$ *Increasing* $\$\leq_v\ \$\leq_v f$
$\Rightarrow (\forall X\bullet X \subseteq FChainU\ f\ \$\leq_v \Rightarrow$ *LinearOrder* $(X, \$\leq_v))$

ccrpou_fchainu_lemma2c =
$\vdash \forall \$\leq_v f$
• *CcRpoU* $\$\leq_v \wedge$ *Increasing* $\$\leq_v\ \$\leq_v f$
$\Rightarrow (\forall x\ y$
• $x \in FChainU\ f\ \$\leq_v \wedge y \in FChainU\ f\ \$\leq_v \Rightarrow y \leq_v x \vee f\ x \leq_v y)$

We now pause to prove the course of values induction principle, deferred to this point because this formulation requires the above result (though this induction principle is not used in the fixedpoint proof which predates it).

It is interesting perhaps that this is the first of the principles for induction over *FChain*s which depends upon the monoticity of the function f which generates the chain.

$$
\begin{array}{l}
\boldsymbol{fchainu\_induction\_thm2} = \\
\quad \vdash \forall\ r\ f\ p \\
\quad\quad \bullet\ CcRpoU\ r \\
\quad\quad\quad\quad \wedge\ Increasing\ r\ r\ f \\
\quad\quad\quad\quad \wedge\ (\forall\ x \\
\quad\quad\quad\quad \bullet\ x \in FChainU\ f\ r \wedge (\forall\ y \bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y \Rightarrow p\ y) \\
\quad\quad\quad\quad\quad\quad \Rightarrow p\ x) \\
\quad\quad\quad\quad \Rightarrow (\forall\ x \bullet\ x \in FChainU\ f\ r \Rightarrow p\ x)
\end{array}
$$

Now we return to the proof of the fixedpoint theorem.

$$
\begin{array}{l}
\boldsymbol{ccrpo\_fixp\_lemma17} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\quad\quad \bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f \\
\quad\quad\quad\quad \wedge\ Rpo\ (Universe,\ \$\leq_v) \\
\quad\quad\quad\quad \wedge\ FChainClosed\ f\ (X,\ \$\leq_v) \\
\quad\quad\quad\quad \Rightarrow (\exists\ e \\
\quad\quad\quad\quad \bullet\ IsLub\ \$\leq_v\ (FChain\ f\ (X,\ \$\leq_v))\ e \\
\quad\quad\quad\quad\quad\quad \wedge\ e = f\ e \\
\quad\quad\quad\quad\quad\quad \wedge\ e \in FChain\ f\ (X,\ \$\leq_v))
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{ccrpo\_fixp\_lemma18} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\quad\quad \bullet\ Increasing\ \$\leq_v\ \$\leq_v\ f \\
\quad\quad\quad\quad \wedge\ Rpo\ (Universe,\ \$\leq_v) \\
\quad\quad\quad\quad \wedge\ FChainClosed\ f\ (X,\ \$\leq_v) \\
\quad\quad\quad\quad \Rightarrow (\exists\ l \bullet\ IsLub\ \$\leq_v\ (FChain\ f\ (X,\ \$\leq_v))\ l \wedge IsLfp\ \$\leq_v\ f\ l)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{ccrpou\_fchainu\_lfp\_lemma} = \\
\quad \vdash \forall\ X\ \$\leq_v\ f \\
\quad\quad \bullet\ CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f \\
\quad\quad\quad\quad \Rightarrow (\exists\ l \bullet\ IsLub\ \$\leq_v\ (FChainU\ f\ \$\leq_v)\ l \wedge IsLfp\ \$\leq_v\ f\ l)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{ccrpo\_fixp\_lemma19} = \\
\quad \vdash \forall\ r\ f \bullet\ CcRpo\ (Universe,\ r) \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ e \bullet\ IsLfp\ r\ f\ e)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{ccrpou\_fixp\_induction\_thm} = \\
\quad \vdash \forall\ r\ f\ p \\
\quad\quad \bullet\ CcRpoU\ r \\
\quad\quad\quad\quad \wedge\ Increasing\ r\ r\ f
\end{array}
$$

$$
\begin{array}{l}
\quad \wedge\ (\forall\ x\bullet\ p\ x \Rightarrow p\ (f\ x)) \\
\quad \wedge\ (\forall\ s \\
\quad \bullet\ (\forall\ x\bullet\ x \in s \Rightarrow p\ x) \wedge \mathit{LinearOrder}\ (s,\ r) \\
\qquad \Rightarrow (\exists\ y\bullet\ p\ y \wedge \mathit{IsLub}\ r\ s\ y)) \\
\quad \Rightarrow p\ (\mathit{Lfp}_c\ r\ f)
\end{array}
$$

The course of values induction theorem over fchains which I proved now seems to me to be more pretty than useful. It seems unlikely that one will not want to prove the induction hypothesis separately for elements which are successors and elements which are limits.

At the least we also need a suitable cases theorem:

$$
\begin{array}{l}
\boldsymbol{fchainu\_cases\_lemma} = \\
\quad \vdash \forall\ r\ f \\
\quad \bullet\ \mathit{CcRpoU}\ r \wedge \mathit{Increasing}\ r\ r\ f \\
\qquad \Rightarrow (\forall\ x \\
\qquad \bullet\ x \in \mathit{FChainU}\ f\ r \\
\qquad\quad \Rightarrow (\exists\ y\bullet\ y \in \mathit{FChainU}\ f\ r \wedge x = f\ y) \\
\qquad\qquad \vee\ x = \mathit{Lub}\ r\ \{z | z \in \mathit{FChainU}\ f\ r \wedge r\ z\ x \wedge \neg\ r\ x\ z\}) \\
\\
\boldsymbol{fchainu\_cases\_lemma2} = \\
\quad \vdash \forall\ r\ f \\
\quad \bullet\ \mathit{CcRpoU}\ r \wedge \mathit{Increasing}\ r\ r\ f \\
\qquad \Rightarrow (\forall\ x \\
\qquad \bullet\ x \in \mathit{FChainU}\ f\ r \\
\qquad\quad \Rightarrow (\exists\ y\bullet\ y \in \mathit{FChainU}\ f\ r \wedge \neg\ y = x \wedge x = f\ y) \\
\qquad\qquad \vee\ x = \mathit{Lub}\ r\ \{z | z \in \mathit{FChainU}\ f\ r \wedge r\ z\ x \wedge \neg\ r\ x\ z\})
\end{array}
$$

If this is indeed the normal way of doing such proofs then the following induction principle, though more complicated than the plain "course of values" principle is probably going to be more convenient.

$$
\begin{array}{l}
\boldsymbol{fchainu\_induction\_thm3} = \\
\quad \vdash \forall\ r\ f\ p \\
\quad \bullet\ \mathit{CcRpoU}\ r \\
\qquad \wedge\ \mathit{Increasing}\ r\ r\ f \\
\qquad \wedge\ (\forall\ x \\
\qquad \bullet\ x \in \mathit{FChainU}\ f\ r \\
\qquad\quad \wedge\ (\forall\ y\bullet\ y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \wedge \neg\ r\ (f\ x)\ y \Rightarrow p\ y) \\
\qquad\quad \Rightarrow p\ (f\ x)) \\
\qquad \wedge\ (\forall\ x \\
\qquad \bullet\ x \in \mathit{FChainU}\ f\ r \\
\qquad\quad \wedge\ x = \mathit{Lub}\ r\ \{y | y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y\} \\
\qquad\quad \wedge\ (\forall\ y\bullet\ y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y \Rightarrow p\ y) \\
\qquad\quad \Rightarrow p\ x) \\
\qquad \Rightarrow (\forall\ x\bullet\ x \in \mathit{FChainU}\ f\ r \Rightarrow p\ x) \\
\\
\boldsymbol{fchainu\_induction\_thm4} = \\
\quad \vdash \forall\ r\ f\ p
\end{array}
$$

$$
\begin{array}{l}
\bullet\ \mathit{CcRpoU}\ r \\
\quad \wedge\ \mathit{Increasing}\ r\ r\ f \\
\quad \wedge\ (\forall\ x \\
\quad \bullet\ x \in \mathit{FChainU}\ f\ r \wedge (\forall\ y \bullet\ y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \Rightarrow p\ y) \\
\qquad \Rightarrow p\ (f\ x)) \\
\quad \wedge\ (\forall\ x \\
\quad \bullet\ x \in \mathit{FChainU}\ f\ r \\
\qquad \wedge\ x = \mathit{Lub}\ r\ \{y | y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y\} \\
\qquad \wedge\ (\forall\ y \bullet\ y \in \mathit{FChainU}\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y \Rightarrow p\ y) \\
\quad \Rightarrow p\ x) \\
\Rightarrow (\forall\ x \bullet\ x \in \mathit{FChainU}\ f\ r \Rightarrow p\ x)
\end{array}
$$

5 CHAIN CO-COMPLETENESS

5.1 Inverse Relations

In order to exploit the duality between induction and co-induction it is useful to have some results about the inverses of relations.

First the definition.

HOL Constant

$$
\begin{array}{l}
\boldsymbol{\mathit{RelInv}} : ('a \to 'a \to \mathit{BOOL}) \to ('a \to 'a \to \mathit{BOOL}) \\
\hline
\forall\ r \bullet\ \mathit{RelInv}\ r = \lambda x\ y \bullet\ r\ y\ x
\end{array}
$$

Then we prove lemmas about various properties of relations and their inverses.

$$
\begin{array}{l}
\boldsymbol{\mathit{refl\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{Refl}\ (X, \mathit{RelInv}\ r) = \mathit{Refl}\ (X, r)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{\mathit{antisym\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{Antisym}\ (X, \mathit{RelInv}\ r) = \mathit{Antisym}\ (X, r)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{\mathit{trans\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{Trans}\ (X, \mathit{RelInv}\ r) = \mathit{Trans}\ (X, r)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{\mathit{trans\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{Trans}\ (X, \mathit{RelInv}\ r) = \mathit{Trans}\ (X, r)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{\mathit{trich\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{Trich}\ (X, \mathit{RelInv}\ r) = \mathit{Trich}\ (X, r)
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{\mathit{linearorder\_inverse\_lemma}} = \\
\quad \vdash \forall\ X\ r \bullet\ \mathit{LinearOrder}\ (X, r) = \mathit{LinearOrder}\ (X, \mathit{RelInv}\ r)
\end{array}
$$

rpo_inverse_lemma =

$\vdash \forall\ X\ r \bullet\ Rpo\ (X,\ RelInv\ r) = Rpo\ (X,\ r)$

isub_inverse_lemma =

$\vdash \forall\ X\ r\ x \bullet\ IsUb\ (RelInv\ r)\ X\ x = IsLb\ r\ X\ x$

islb_inverse_lemma =

$\vdash \forall\ X\ r\ x \bullet\ IsLb\ (RelInv\ r)\ X\ x = IsUb\ r\ X\ x$

islub_inverse_lemma =

$\vdash \forall\ X\ r\ x \bullet\ IsLub\ (RelInv\ r)\ X\ x = IsGlb\ r\ X\ x$

isglb_inverse_lemma =

$\vdash \forall\ X\ r\ x \bullet\ IsGlb\ (RelInv\ r)\ X\ x = IsLub\ r\ X\ x$

islfp_inverse_lemma =

$\vdash \forall\ r\ f\ e \bullet\ IsLfp\ (RelInv\ r)\ f\ e = IsGfp\ r\ f\ e$

isgfp_inverse_lemma =

$\vdash \forall\ r\ f\ e \bullet\ IsGfp\ (RelInv\ r)\ f\ e = IsLfp\ r\ f\ e$

increasing_inverse_lemma =

$\vdash \forall\ r1\ r2 \bullet\ Increasing\ (RelInv\ r1)\ (RelInv\ r2) = Increasing\ r1\ r2$

5.2 Chain Co-Complete Partial Orders

The results in this section could be achieved in either of two ways. The first is by systematic modifications to the proof above to obtain their duals. The second is by use of the original theorems on inverses of relationship. I have done some by the first method, which is a bit tedious. To use the second method here is the definition of the inverse of a relationship.

HOL Constant

ChainCoComplete : $('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$\forall\ X\ r \bullet\ ChainCoComplete\ (X,\ r) \Leftrightarrow \forall Y \bullet\ Y \subseteq X \wedge LinearOrder\ (Y,\ r) \Rightarrow$
$\quad \exists x \bullet\ x \in X \wedge IsGlb\ r\ Y\ x$

chaincocomplete_dual_lemma =

$\vdash \forall\ X\ r \bullet\ ChainCoComplete\ (X,\ r) = ChainComplete\ (X,\ RelInv\ r)$

HOL Constant

CoCcRpo : $('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$\forall\ r \bullet\ CoCcRpo\ r \Leftrightarrow Rpo\ r \wedge ChainCoComplete\ r$

coccrpo_dual_lemma =
$\vdash \forall\ X\ r\bullet\ CoCcRpo\ (X,\ r) = CcRpo\ (X,\ RelInv\ r)$

HOL Constant
CoCcRpoU : $('a \to 'a \to BOOL) \to BOOL$

$\forall\ r\bullet\ CoCcRpoU\ r \Leftrightarrow CoCcRpo\ (Universe,\ r)$

coccrpou_dual_lemma =
$\vdash \forall\ r\bullet\ CoCcRpoU\ r = CcRpoU\ (RelInv\ r)$

HOL Constant
FCoChainClosed : $('a \to 'a) \to ('a\ SET \times ('a \to 'a \to BOOL)) \to BOOL$

$\forall\ f\ X\ r\bullet\ FCoChainClosed\ f\ (X,\ r) \Leftrightarrow$
$\quad FClosed\ f\ X \wedge ChainCoComplete\ (X,\ r)$

fcochainclosed_dual_lemma =
$\vdash \forall\ f\ X\ r\bullet\ FCoChainClosed\ f\ (X,\ r) = FChainClosed\ f\ (X,\ RelInv\ r)$

HOL Constant
FCoChain : $('a \to 'a) \to ('a\ SET \times ('a \to 'a \to BOOL)) \to 'a\ SET$

$\forall\ f\ X\ r\bullet\ FCoChain\ f\ (X,\ r) = \bigcap \{Y \mid Y \subseteq X \wedge FCoChainClosed\ f\ (Y,\ r)\}$

fcochain_dual_lemma =
$\vdash \forall\ X\ r\ f\bullet\ FCoChain\ f\ (X,\ r) = FChain\ f\ (X,\ RelInv\ r)$

fcochain_lemma1 =
$\vdash \forall\ X\ r\ f\bullet\ FCoChainClosed\ f\ (X,\ r) \Rightarrow FCoChain\ f\ (X,\ r) \subseteq X$

fcochain_fcochainclosed_lemma =
$\vdash \forall\ X\ r\ f$
$\bullet\ Antisym\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\quad \Rightarrow FCoChainClosed\ f\ (FCoChain\ f\ (X,\ r),\ r)$

fcochain_fcochainclosed_lemma2 =
$\vdash \forall\ X\ r\ f$
$\bullet\ Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\quad \Rightarrow FCoChainClosed\ f\ (FCoChain\ f\ (X,\ r),\ r)$

fcoc_fcocclosed_fcoc_lemma =
$\vdash \forall\ X\ r\ f$
$\bullet\ Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$

$\Rightarrow (\forall\ x\bullet\ x \in FCoChain\ f\ (X,\ r) \Rightarrow f\ x \in FCoChain\ f\ (X,\ r))$

$fcochain\_lemma2$ =
$\vdash \forall\ X\ \ Y\ r\ f$
• $Rpo\ (Universe,\ r)$
$\wedge\ FCoChainClosed\ f\ (X,\ r)$
$\wedge\ Y \subseteq FCoChain\ f\ (X,\ r)$
$\wedge\ FCoChainClosed\ f\ (Y,\ r)$
$\Rightarrow\ Y = FCoChain\ f\ (X,\ r)$

$coccrpo\_fixp\_lemma1$ =
$\vdash \forall\ X\ r\ f$
• $Increasing\ r\ r\ f$
$\Rightarrow FClosed\ f\ \{x|x \in FCoChain\ f\ (X,\ r) \wedge (r\ (f\ x)\ x \vee x = f\ x)\}$

$coccrpo\_fixp\_lemma2$ =
$\vdash \forall\ X\ r\ f$
• $Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\Rightarrow ChainCoComplete$
$(\{x|x \in FCoChain\ f\ (X,\ r) \wedge (r\ (f\ x)\ x \vee x = f\ x)\},\ r)$

$coccrpo\_fixp\_lemma3$ =
$\vdash \forall\ X\ r\ f$
• $Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\Rightarrow FCoChainClosed$
f
$(\{z|z \in FCoChain\ f\ (X,\ r) \wedge (r\ (f\ z)\ z \vee z = f\ z)\},\ r)$

$coccrpo\_fixp\_lemma4$ =
$\vdash \forall\ X\ r\ f$
• $Increasing\ r\ r\ f \wedge Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\Rightarrow (\forall\ z\bullet\ z \in FCoChain\ f\ (X,\ r) \Rightarrow r\ (f\ z)\ z \vee z = f\ z)$

$coccrpo\_fixp\_lemma18$ =
$\vdash \forall\ X\ \$\leq_v\ f$
• $Increasing\ \$\leq_v\ \$\leq_v\ f$
$\wedge\ Rpo\ (Universe,\ \$\leq_v)$
$\wedge\ FCoChainClosed\ f\ (X,\ \$\leq_v)$
$\Rightarrow (\exists\ l\bullet\ IsGlb\ \$\leq_v\ (FCoChain\ f\ (X,\ \$\leq_v))\ l \wedge IsGfp\ \$\leq_v\ f\ l)$

$coccrpo\_fixp\_lemma19$ =
$\vdash \forall\ r\ f\bullet\ CoCcRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ e\bullet\ IsGfp\ r\ f\ e)$

5.3 Closure

If you start from bottom and iterate a function over some complete partial order you get a least fixed point of the function. If you start somewhere else you get a "closure". From a monotone function you obtain by transfinite iteration a closure operator.

The purpose of this subsection is to adapt the above material so that it can be applied to closure operators obtained in this way.

For this to work we need only chain completeness, so we will do the theory first in that context.

HOL Constant

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

$$\forall\ \$\leq_v\ f\ x\bullet\ Closed_c\ (\$\leq_v, f)\ x \Leftrightarrow f\ x \leq_v x$$

HOL Constant

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

$$\forall\ \$\leq_v\ f\ x\bullet\ Closure_c\ (\$\leq_v, f)\ x = Glb\ \$\leq_v\ \{y \mid x \leq_v y \wedge Closed_c\ (\$\leq_v, f)\ y\}$$

HOL Constant

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

$$\forall\ \$\leq_v\ f\ x\bullet\ Closure_d\ (\$\leq_v, f)\ x = Lub\ \$\leq_v\ (FChain\ f\ (\{y \mid x \leq_v y\}, \$\leq_v))$$

The first result I am looking for is that the closure of a function f maps each element x in its domain to the least fixed point of f which is greater than x.

6 COMPLETE PARTIAL ORDERS

HOL Constant

$$\boldsymbol{CRpoU}: ('a \to 'a \to BOOL) \to BOOL$$

$$\forall\ r\bullet\ CRpoU\ r \Leftrightarrow RpoU\ r \wedge LubsExist\ r \wedge GlbsExist\ r$$

$$
\begin{array}{l}
\textbf{\textit{crpou\_ccrpou\_lemma}} = \\
\quad \vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow CcRpoU\ r \\
\\
\textbf{\textit{crpou\_lub\_glb\_}}\varnothing\textbf{\textit{\_lemma}} = \\
\quad \vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ x\bullet\ r\ (Lub\ r\ \{\})\ x \wedge r\ x\ (Glb\ r\ \{\})) \\
\\
\textbf{\textit{crpou\_fc\_clauses}} = \\
\quad \vdash \forall\ r \\
\qquad \bullet\ CRpoU\ r \\
\qquad\quad \Rightarrow GlbsExist\ r \\
\qquad\qquad \wedge\ LubsExist\ r \\
\qquad\qquad \wedge\ (\forall\ x\bullet\ r\ x\ x) \\
\qquad\qquad \wedge\ (\forall\ x\ y\ z\bullet\ r\ x\ y \wedge r\ y\ z \Rightarrow r\ x\ z) \\
\qquad\qquad \wedge\ (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y) \\
\\
\textbf{\textit{crpou\_glb\_lfp\_lemma1}} = \\
\quad \vdash \forall\ r\ f \\
\qquad \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \\
\qquad\quad \Rightarrow (\exists\ e\bullet\ IsGlb\ r\ \{x|r\ (f\ x)\ x\}\ e \wedge IsLfp\ r\ f\ e) \\
\\
\textbf{\textit{crpou\_increasing\_lfp\_lemma1}} = \\
\quad \vdash \forall\ r\ f\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ l\bullet\ IsLfp\ r\ f\ l) \\
\\
\textbf{\textit{crpou\_increasing\_lfp\_lemma2}} = \\
\quad \vdash \forall\ r\ f\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow IsLfp\ r\ f\ (Lfp_c\ r\ f) \\
\\
\textbf{\textit{crpou\_lub\_gfp\_lemma1}} = \\
\quad \vdash \forall\ r\ f \\
\qquad \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \\
\qquad\quad \Rightarrow (\exists\ e\bullet\ IsLub\ r\ \{x|r\ x\ (f\ x)\}\ e \wedge IsGfp\ r\ f\ e) \\
\\
\textbf{\textit{crpou\_increasing\_gfp\_lemma1}} = \\
\quad \vdash \forall\ r\ f\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ g\bullet\ IsGfp\ r\ f\ g) \\
\\
\textbf{\textit{crpou\_increasing\_gfp\_lemma2}} = \\
\quad \vdash \forall\ r\ f\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow IsGfp\ r\ f\ (Gfp_c\ r\ f) \\
\\
\textbf{\textit{isglb\_glb\_crpou\_lemma}} = \\
\quad \vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ X\ z\bullet\ IsGlb\ r\ X\ z \Rightarrow Glb\ r\ X = z) \\
\\
\textbf{\textit{islub\_lub\_crpou\_lemma}} = \\
\quad \vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ X\ z\bullet\ IsLub\ r\ X\ z \Rightarrow Lub\ r\ X = z)
\end{array}
$$

6.1 Continuity and Induction

To prove the properties of least fixed points (not the properties of their members) one approach is to prove:

- that the bottom element has the property
- that the function preserves the property
- that the property is "continuous"

The purpose of this section is to formulate suitable notions of continuity.

I did consider using the theory "topology" but that seemed likely to be too difficult to bend to this context.

Here's an induction principle:

crpou_induction_thm =
$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge (\forall\ s \bullet (\forall\ x \bullet\ x \in s \Rightarrow p\ x) \Rightarrow p\ (Lub\ r\ s)) \Rightarrow p\ (Lfp_c\ r\ f)$$

Here's a stronger induction principle:

crpou_induction_thm2 =
$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge (\forall\ s \bullet\ LinearOrder\ (s, r) \wedge (\forall\ x \bullet\ x \in s \Rightarrow p\ x) \Rightarrow p\ (Lub\ r\ s)) \Rightarrow p\ (Lfp_c\ r\ f)$$

I'm going to give a name to the condition required in this induction principle.

HOL Constant

ContProp: $('a \to 'a \to BOOL) \to ('a \to BOOL) \to BOOL$

$$\forall\ r\ p \bullet\ ContProp\ r\ p \Leftrightarrow \forall s \bullet (\forall y \bullet\ y \in s \Rightarrow p\ y) \Rightarrow p\ (Lub\ r\ s)$$

And then restate the principle using the name:

contprop_induction_thm =
$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge ContProp\ r\ p \Rightarrow p\ (Lfp_c\ r\ f)$$

Least fixed point induction using a notion of continuity which is independent of the functor about whose least fixed point we are reasoning now seems to me a poor idea (not least because I have now a problem for which it is obviously inadequate).

HOL Constant

$$\textbf{\textit{Rising}}: ('a \to 'a) \to ('a \to 'a \to BOOL) \to ('a \to BOOL)$$

$$\forall\ f\ r\bullet\ Rising\ f\ r = \lambda x\bullet\ r\ x\ (f\ x)$$

HOL Constant

$$\textbf{\textit{Falling}}: ('a \to 'a) \to ('a \to 'a \to BOOL) \to ('a \to BOOL)$$

$$\forall\ f\ r\bullet\ Falling\ f\ r = \lambda x\bullet\ r\ (f\ x)\ x$$

A second notion of continuity is:

HOL Constant

$$\textbf{\textit{ContProp3}}: ('a \to 'a) \to ('a \to 'a \to BOOL) \to ('a \to BOOL) \to BOOL$$

$$\forall\ f\ r\ p\bullet\ ContProp3\ f\ r\ p \Leftrightarrow \forall x\bullet\ (\forall y\bullet\ Rising\ f\ r\ y \wedge r\ y\ x \wedge \neg\ r\ x\ y \Rightarrow p\ y) \Rightarrow p\ x$$

Hopefully all properties p which have this property are true of the least fixed point of f relative to r.

$$?\ \forall r\bullet\ CRpoU\ r \Rightarrow \forall f\ p\bullet\ ContProp3\ f\ r\ p \Rightarrow p\ (Lfp_c\ r\ f)$$

7 INDUCTIVE DEFINITIONS OF SETS

We begin with inductive definitions of sets, later addressing the conversion of these sets into types.

The simplest example of interest here is the natural numbers, which can be defined (in HOL) as the smallest set of individuals which includes zero (the individual which is not in the range of the one-one function whose existence is asserted by the usual axiom of infinity) and is closed under the successor function (which is that same one-one function).

We can think of this as forming the natural numbers by starting with some set (0) and then adding further elements following some prescription until no more can be added. Because we are always adding values, the operation on the set-of-values-so-far is monotonic. If the closure is supplied in a suitable manner then a completely general proof of monotonicity will suffice.

There is a little difficulty in doing this automatically because the operators under which closure is wanted (counting the starting points as 0-ary operators) will be of diverse types.

We keep the constructor exactly as it is required on the representation type. This is combined with an "immediate content" function on the domain of the constructor to give a relation which indicates which values are immediate constituents of a constructed value, and then we close up the empty set on the principle of adding a constructed value whenever its immediate constituents are available.

In addition to the constructor function and the content information we want to allow some constraint on values which are acceptable for the construction so that it need not be defined over the entire representation type. In fact this can be coded into the content function by making it reflexive for values which we wish to exclude from the domain. Actually its type doesn't allow reflexive, but mapping these to the universe of the representation type will do the trick.

7.1 Generalised Hereditarily-P Sets

The notion of a set being "hereditarily P" for some property of sets P is to be generalised in the following two ways:

1. Rather than taking any fixed notion of set, the machinery is to work in as broad a range of "membership structures" as possible, the minimal conditions on membership structures for this to work will be worked out as we go.

2. To allow more complex constructions, the generalised "hereditarily" will be parameterised by something more complex than a simple property, in particular it is not to be required that all the members of a "hereditarily X" set will be "hereditarily X", but only the bona-fide "constituents". For example, in the notion of a hereditarily pure function, it is not the members of a hereditarily pure function which must be hereditarily pure functions, rather, the elements of its domain and range.

7.2 Hereditarily Over a Function

Parameterising the notion of hereditarily by a relation allows the notion to be relativised to an arbitrary set theory or pseudo-set-theory. As well as permitting the notion to be used within exotic set theories such as NF, the generalised notion can be applied to domains like the natural numbers, which given a suitable relation will masquerade as hereditarily finite sets, and can be further refined by hereditarily-p notions.

However, the presumption still present is that, albeit with a generalised notion of membership, the kinds construction involved are simply set formation.

It is easy to find natural examples which do not fit into this scheme. For example the hereditarily pure functions. If one simply takes the property P of being a pure function, then Hereditarily-P does not give what is desired, because the formation of a pure function over pure functions involves a construction more complex than set formation (or at least, requires multiple set formations, firstly the formation of ordered pairs and then the collection of the ordered pairs into a set).

To allow for this kind of generalisation we must leave behind the idea that the construction is taking place in the domain of a set theory.

We do this by working from a function which models the available constructions or derivations by mapping a set of objects (which need not themselves be sets) to the set of new objects which can be constructed from those objects. Other things might be used in the construction, but no other "objects", so the purpose of this map, which I will call a content function, is not to fully capture the details of the available constructors or derivation rules, but rather to capture the *content* of the available constructions or the premises of the rules.

From a content function a fixed point can be obtained. This is done by converting it into a monotonic function and then using the Knaster-Tarski result.

HOL Constant

$$\boldsymbol{Fun2MonoFun}: ('a\ SET \to 'a\ SET) \to ('a\ SET \to 'a\ SET)$$

$$\forall\ f\ s\bullet\ Fun2MonoFun\ f\ s = \{x \mid \exists t\bullet\ t \subseteq s \wedge x \in f\ t\}$$

This function does always deliver a monotonic result:

$\vert$ ***F2MF_Monotonic_thm*** $=$
$\vert$ $\quad \vdash \forall\ r\bullet\ Monotonic\ (Fun2MonoFun\ f)$

We now define the function which maps such a content function to the least fixed point of the monotonic function obtained from it.

HOL Constant

$\vert$ ***HeredFun***: $('a\ SET \to 'a\ SET) \to 'a\ SET$
$\vert$ ———————————————————————
$\vert$ $\forall\ f\bullet\ HeredFun\ f = Lfp\ (Fun2MonoFun\ f)$

And its dual:

HOL Constant

$\vert$ ***CoHeredFun***: $('a\ SET \to 'a\ SET) \to 'a\ SET$
$\vert$ ———————————————————————
$\vert$ $\forall\ f\bullet\ CoHeredFun\ f = Gfp\ (Fun2MonoFun\ f)$

We define appropriate notions of closure and co-closure for expressing the key properties of these sets.

SML

$\vert$ *declare_infix* $(300,$ "*ClosedUnderFun*");
$\vert$ *declare_infix* $(300,$ "*OpenUnderFun*");

HOL Constant

$\vert$ \$***ClosedUnderFun*** : $'a\ SET \to ('a\ SET \to 'a\ SET) \to BOOL$
$\vert$ ———————————————————————
$\vert$ $\forall s\ f\bullet\ s\ ClosedUnderFun\ f \Leftrightarrow s\ ClosedUnder\ (Fun2MonoFun\ f)$

HOL Constant

$\vert$ \$***OpenUnderFun*** : $'a\ SET \to ('a\ SET \to 'a\ SET) \to BOOL$
$\vert$ ———————————————————————
$\vert$ $\forall s\ f\bullet\ s\ OpenUnderFun\ f \Leftrightarrow s\ OpenUnder\ (Fun2MonoFun\ f)$

We then prove the obvious claims about closure and openness, and the simplest induction and co-induction principles.

$\vert$ ***ClosedUnderFun_thm*** $=$
$\vert$ $\quad \vdash \forall\ f\ s\bullet\ s\ ClosedUnderFun\ f \Leftrightarrow (\forall\ t\bullet\ t \subseteq s \Rightarrow f\ t \subseteq s)$
$\vert$ ***OpenUnderFun_thm1*** $=$
$\vert$ $\quad \vdash \forall\ f\ s\bullet\ s\ OpenUnderFun\ f$
$\vert$ $\qquad \Leftrightarrow (\forall\ t\bullet\ s \subseteq t \Rightarrow (\forall\ e\bullet\ e \in s \Rightarrow (\exists\ t'\bullet\ t' \subseteq t \wedge e \in f\ t')))$
$\vert$ ***OpenUnderFun_thm2*** $=$
$\vert$ $\quad \vdash \forall\ f\ s\bullet\ s\ OpenUnderFun\ f \Leftrightarrow (\forall\ t\bullet\ s \subseteq t \Rightarrow s \subseteq Fun2MonoFun\ f\ t)$

$$
\begin{array}{l}
\textbf{\textit{HeredFun\_Closed\_thm}} = \\
\quad \vdash \forall f\bullet\ (\mathit{HeredFun}\ f)\ \mathit{ClosedUnderFun}\ f \\
\textbf{\textit{HeredFun\_Closed\_thm1}} = \\
\quad \vdash \forall\ f\ x\bullet\ (\exists\ t\bullet\ (\forall\ x\bullet\ x \in t \Rightarrow x \in \mathit{HeredFun}\ f) \wedge x \in f\ t) \\
\qquad \Rightarrow x \in \mathit{HeredFun}\ f \\
\\
\textbf{\textit{HeredFun\_Open\_thm}} = \\
\quad \vdash \forall f\bullet\ (\mathit{HeredFun}\ f)\ \mathit{OpenUnderFun}\ f \\
\textbf{\textit{HeredFun\_Open\_thm1}} = \\
\quad \vdash \forall\ f\ x\bullet\ x \in \mathit{HeredFun}\ f \\
\quad \Rightarrow (\exists\ t\bullet\ (\forall\ x\bullet\ x \in t \Rightarrow x \in \mathit{HeredFun}\ f) \wedge x \in f\ t) \\
\\
\textbf{\textit{HeredFun\_induction\_thm}} = \\
\quad \vdash \forall f\ s\bullet\ s\ \mathit{ClosedUnderFun}\ f \Rightarrow \mathit{HeredFun}\ f \subseteq s \\
\textbf{\textit{HeredFun\_induction\_thm1}} = \\
\quad \vdash \forall f\ s\bullet\ (\forall t\bullet\ t \subseteq s \Rightarrow f\ t \subseteq s) \Rightarrow \mathit{HeredFun}\ f \subseteq s
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{CoHeredFun\_Closed\_thm}} = \\
\quad \vdash \forall f\bullet\ (\mathit{CoHeredFun}\ f)\ \mathit{ClosedUnderFun}\ f \\
\textbf{\textit{CoHeredFun\_Open\_thm}} = \\
\quad \vdash \forall f\bullet\ (\mathit{CoHeredFun}\ f)\ \mathit{OpenUnderFun}\ f \\
\\
\textbf{\textit{CoHeredFun\_coinduction\_thm}} = \\
\quad \vdash \forall s\ f\bullet\ s\ \mathit{OpenUnderFun}\ f \Rightarrow s \subseteq \mathit{CoHeredFun}\ f \\
\textbf{\textit{CoHeredFun\_coinduction\_thm2}} = \\
\quad \vdash \forall\ s\ f\bullet\ s \subseteq \mathit{Fun2MonoFun}\ f\ s \Rightarrow s \subseteq \mathit{CoHeredFun}\ f
\end{array}
$$

The following stronger induction and co-induction principles correspond to the notion of strong induction principle in the hol4 package for definition of relations by taking transitive closures. These principles are not specific to the definition of relations but their inclusion is motivated by problems arising in the definition of relations, more specifically in obtaining Church-Rosser or confluence results in combinatory logics.

The following principle is a very slight strengthening of the basic induction principle:

$$
\begin{array}{l}
\textbf{\textit{HeredFun\_induct\_thm0}} = \\
\quad \vdash \forall f\ s\bullet\ (s \cap \mathit{HeredFun}\ f)\ \mathit{ClosedUnderFun}\ f \Rightarrow \mathit{HeredFun}\ f \subseteq s
\end{array}
$$

7.3 Collections of Rules

When I came to try out the above (see [4]) I concluded that the most convenient kind of object to define is a *set of rules*, in which each rule is a set of premises and a single conclusion. This is a relation which need not be functional and need not be one-one (there can be several conclusions from the same set of premises, and several distinct sets of premises which yield the same conclusion).

So I am defining here how to convert one of those things into a content function.

HOL Constant

$$\textbf{\textit{Rules2Fun}}: ('a\ SET \times 'a)\ SET \to ('a\ SET \to 'a\ SET)$$

$$\forall\ r \bullet\ Rules2Fun\ r = \lambda s \bullet \{x \mid (s,\ x) \in r\}$$

Rules2Fun is a bijection and so could be inverted.

7.4 Hereditarily Over a Relation

An alternative, but less general approach is to define the constructor from a 'content' relation which indicates when a value is an immediate constituent of another value. This approach is less general because for some inductive definitions of sets, e.g. for the set of theorems of a formal system, there are several ways of constructing the same object and so no obvious content relation which can be used to define the set.

HOL Constant

$$\textbf{\textit{Rel2Fun}}: ('a \to 'a \to BOOL) \to ('a\ SET \to 'a\ SET)$$

$$\forall\ r \bullet\ Rel2Fun\ r = \lambda s \bullet \{x \mid \{y \mid r\ y\ x\} = s\}$$

We now define the function which maps a content relation to the least fixed point of the monotonic function obtained from it.

HOL Constant

$$\textbf{\textit{HeredRel}}: ('a \to 'a \to BOOL) \to 'a\ SET$$

$$\forall\ r \bullet\ HeredRel\ r = HeredFun\ (Rel2Fun\ r)$$

And its dual:

HOL Constant

$$\textbf{\textit{CoHeredRel}}: ('a \to 'a \to BOOL) \to 'a\ SET$$

$$\forall\ r \bullet\ CoHeredRel\ r = CoHeredFun\ (Rel2Fun\ r)$$

To accomplish this we need the concepts of closure and co-closure for sets relative to one of our 'content relations'

SML

```
declare_infix (300, "ClosedUnderRel");
declare_infix (300, "OpenUnderRel");
```

HOL Constant

$$\$\textbf{\textit{ClosedUnderRel}} : 'a\ SET \to ('a \to 'a \to BOOL) \to BOOL$$

$$\forall s\ r \bullet\ s\ ClosedUnderRel\ r \Leftrightarrow s\ ClosedUnderFun\ (Rel2Fun\ r)$$

HOL Constant

$\$\mathbf{OpenUnderRel} : 'a\ SET \to ('a \to 'a \to BOOL) \to BOOL$

$\forall s\ r\bullet\ s\ OpenUnderRel\ r \Leftrightarrow s\ OpenUnderFun\ (Rel2Fun\ r)$

$\mathbf{HeredRel\_Closed\_thm} =$
$\quad \vdash \forall f\bullet\ (HeredRel\ f)\ ClosedUnderRel\ f$
$\mathbf{HeredRel\_Open\_thm} =$
$\quad \vdash \forall f\bullet\ (HeredRel\ f)\ OpenUnderRel\ f$
$\mathbf{HeredRel\_induction\_thm} =$
$\quad \vdash \forall s\ f\bullet\ s\ ClosedUnderRel\ f \Rightarrow HeredRel\ f \subseteq s$

$\mathbf{CoHeredRel\_Closed\_thm} =$
$\quad \vdash \forall f\bullet\ (CoHeredRel\ f)\ ClosedUnderRel\ f$
$\mathbf{CoHeredRel\_Open\_thm} =$
$\quad \vdash \forall f\bullet\ (CoHeredRel\ f)\ OpenUnderRel\ f$
$\mathbf{CoHeredRel\_coinduction\_thm} =$
$\quad \vdash \forall s\ f\bullet\ s\ OpenUnderRel\ f \Rightarrow s \subseteq CoHeredRel\ f$

7.5 Hereditarily Over a Property

We now presume that the construction is always set formation and that the sets which are formed are those satisfying the given property. However, everything is relativised to an arbitrary membership structure.

This is really (only a small generalisation of) what mathematicians call "Hereditarily-P" sets, so I shall give it its proper name.

A content relation is obtained from a property over a membership structure as follows:

HOL Constant

$\mathbf{Prop2Rel}: ('a\ SET \times ('a \to 'a \to BOOL)) \to ('a \to BOOL) \to ('a \to 'a \to BOOL)$

$\forall\ (X, \$<<)\ p\bullet\ Prop2Rel\ (X, \$<<)\ p = \lambda x\ y\bullet\ x << y \wedge p\ y$

HOL Constant

$\mathbf{Hereditary}: ('a\ SET \times ('a \to 'a \to BOOL)) \to ('a \to BOOL) \to 'a\ SET$

$\forall\ (X, \$<<)\ p\bullet\ Hereditary\ (X, \$<<)\ p = HeredRel\ (Prop2Rel\ (X, \$<<)\ p)$

And its dual:

HOL Constant

$\mathbf{CoHereditary}: ('a\ SET \times ('a \to 'a \to BOOL)) \to ('a \to BOOL) \to 'a\ SET$

$\forall\ (X, \$<<)\ p\bullet\ CoHereditary\ (X, \$<<)\ p = CoHeredRel\ (Prop2Rel\ (X, \$<<)\ p)$

To accomplish this we need the concepts of closure and co-closure for sets relative to one of our 'content relations'

HOL Constant

$\$\boldsymbol{ClosedUnderProp} : ('a\ SET \times ('a \to 'a \to BOOL)) \to 'a\ SET \to ('a \to BOOL) \to BOOL$

$\forall(X, \$<<)\ s\ p\bullet\ ClosedUnderProp\ (X, \$<<)\ s\ p \Leftrightarrow s\ ClosedUnderRel\ (Prop2Rel\ (X, \$<<)\ p)$

HOL Constant

$\$\boldsymbol{OpenUnderProp} : ('a\ SET \times ('a \to 'a \to BOOL)) \to 'a\ SET \to ('a \to BOOL) \to BOOL$

$\forall(X, \$<<)\ s\ p\bullet\ OpenUnderProp\ (X, \$<<)\ s\ p \Leftrightarrow s\ OpenUnderRel\ (Prop2Rel\ (X, \$<<)\ p)$

$\boldsymbol{Hereditary\_Closed\_thm} =$
$\vdash \forall(X, \$<<)\ p\bullet$
$ClosedUnderProp\ (X, \$<<)\ (Hereditary\ (X, \$<<)\ p)\ p$

$\boldsymbol{Hereditary\_Open\_thm} =$
$\vdash \forall(X, \$<<)\ p\bullet$
$OpenUnderProp\ (X, \$<<)\ (Hereditary\ (X, \$<<)\ p)\ p$

$\boldsymbol{Hereditary\_induction\_thm} =$
$\vdash \forall(X, \$<<)\ s\ p\bullet$
$ClosedUnderProp\ (X, \$<<)\ s\ p \Rightarrow Hereditary\ (X, \$<<)\ p \subseteq s$

$\boldsymbol{CoHereditary\_Closed\_thm} =$
$\vdash \forall(X, \$<<)\ p\bullet$
$ClosedUnderProp\ (X, \$<<)\ (CoHereditary\ (X, \$<<)\ p)\ p$

$\boldsymbol{CoHereditary\_Open\_thm} =$
$\vdash \forall(X, \$<<)\ p\bullet$
$OpenUnderProp\ (X, \$<<)\ (CoHereditary\ (X, \$<<)\ p)\ p$

$\boldsymbol{CoHereditary\_coinduction\_thm} =$
$\vdash \forall(X, \$<<)\ s\ p\bullet$
$OpenUnderProp\ (X, \$<<)\ s\ p \Rightarrow s \subseteq CoHereditary\ (X, \$<<)\ p$

7.6 Sets Defined Using CCPs

The next two variations on the theme move us closer to the form in which the constructors for an inductively defined set are likely to be presented, and in particular to the kinds of constructors whose definitions can be derived from a *signature* for the desired inductive type. It is particularly relevant to inductive definitions in a type theory rather than a set theory, where the constructors may have diverse types and some work is involved in obtaining a monotonic function whose fixed point can be taken.

There are two cases we consider.

In the most common case the compounded constructor (operating over the disjoint union of mutually defined sets) is an injection, and the content relation is probably the best way to define the type, which turns out to be the field of the well-founded part of the content relation, so that a well-founded relation for inductive proofs comes easily.

In the less common case the compounded constructor is not injective, and cannot therefore be adequately represented by a content relation, so we use a function from content sets to the set of objects with that content (allowing that the same object may appear in more than one of these sets as a result of being constructible by more than one constructor).

A CCP (constructor, content, predicate triple) consisting of:

1. a compounded constructor function
2. a content function
3. a compounded predicate

play an important role here and in the sequel and is therefore defined as a HOL labelled product as follows:

HOL Labelled Product

CCP

$$\begin{array}{l} \boldsymbol{Constructor}: {'b} \to {'a}; \\ \boldsymbol{Content}: {'b} \to {'a}\ SET; \\ \boldsymbol{Predicate}: {'b} \to BOOL \end{array}$$

In this the type variable $'b$ is used for the type of the compounded constructor function which is derived from the type of the desired constructors as follows.

1. the type is the disjoint union of the domain types for each individual constructor
2. for each constructor the domain type is the product of the types of the domains if it is a curried constructor, e.g. for a constructor of type $X \to Y \to Z$ the domain type is $X \times Y$. types to be introduced.

The second type parameter is the representation type over which the constructors are defined, and which will be used to introduce the new types (if required).

A CCP can be converted into either a relation or a function as required for $HeredFun$ or $HeredRel$ using the following functions.

HOL Constant

$$\boldsymbol{CCP2Fun}: ('a,'b)CCP \to ('a\ SET \to {'a}\ SET)$$

$$\forall\ ccp\ x \bullet\ CCP2Fun\ ccp\ x = \{y \mid \exists b \bullet\ y = Constructor\ ccp\ b \wedge x = Content\ ccp\ b \wedge Predicate\ ccp\ b\}$$

HOL Constant

$$\textbf{\textit{CCP2Rel}}: (\prime a, \prime b)\ CCP \to (\prime a \to \prime a \to BOOL)$$

$$\forall\ ccp\ x\ y \bullet\ CCP2Rel\ ccp\ x\ y \Leftrightarrow$$
$$\quad \exists b \bullet\ y = Constructor\ ccp\ b \wedge x \in Content\ ccp\ b \wedge Predicate\ ccp\ b$$

In cases such as that of the "hereditarily pure functions" which fall outside the scope of definition using $Hereditarily$ but are similar in spirit the desired effect can be obtained using a $CCP2$ in which the constructor is the identity function. The predicate in this particular case would be the property of being a many one relation, and the content function would yield the field of the relation. The $CCP2Rel$ conversion and $HeredRel$ would then yield the desired set and an induction principle.

7.7 Using Multiple CCP's

In principle it is possible to obtain a fixed point from a system of constructors via a single CCP in which the type variable $\prime b$ is instantiated to the disjoint union of the domains of the (uncurried) constructors. This however leads, to unnecessarily large types, and to complications in the constructor function which arise from manipulating disjoint unions with many components.

The domain types of the constructors have only a transitory appearance in the process of obtaining a fixed point, as can be seen from the type of the functions $CCP2Fun$ and $CCP2Rel$ which convert a structure in which the type variable $\prime b$ occurs into one in which it does not occur. This suggests that it may be best to convert each constructor individually into a content relation before attempting to compound the constructors and obtain a fixed point.

The details of how this can be done are sensitive to issues which have not yet been considered, relating to the manner in which constructors are compounded. In the most common kind of inductive definition used in computer science it is expected that the constructors will be injective and will have disjoint ranges, resulting in an inductive type which is analogous to an initial or free algebra. In other cases this may not be required.

The following definitions are intended only to address the former case.

In compounding the constructors it is necessary to ensure that the ranges of the constructors are disjoint in the compounded constructor, and also to ensure where there is more than one type that the types are disjoint, and that only objects of the appropriate 'type' are supplied to the individual constructors in the compounded constructor. The disjointness of the constructors is to be achieved by some kind of tagging operation in which the value yielded by a constructor is tagged with a value unique to that constructor. This would also suffice to make the types disjoint, but it seems more convenient to use a double tagging to achieve the desired effect. First the value is tagged by a number which discriminates between the constructors yielding values of a type, and then it is tagged again with the type.

This whole process can be accomplished by a single function application to a list of lists of constructor functions or relations, parameterised by an application specific tagging operation.

HOL Constant

$$\textbf{\textit{MapNFun}}: (\mathbb{N} \to \prime a \to \prime a) \to \prime a\ LIST \to \prime a\ LIST$$

$$\forall\ tf\ al \bullet\ MapNFun\ tf\ al = Map\ (Uncurry\ tf)\ (Combine\ (1\ ..{}_L\ (Length\ al))\ al)$$

HOL Constant

$\boldsymbol{MapTag}$: $(\mathbb{N} \to 'a \to 'a) \to ('a \to 'a)\ LIST \to ('a \to 'a)\ LIST$

$\forall\ tf\ al \bullet\ MapTag\ tf\ al = MapNFun\ (\lambda n\ f\ a \bullet\ tf\ n\ (f\ a))\ al$

HOL Constant

$\boldsymbol{LiftTag}$: $(\mathbb{N} \to 'a \to 'a) \to (\mathbb{N} \to 'a\ SET \to 'a\ SET)$

$\forall\ tf\ n\ s \bullet\ LiftTag\ tf\ n\ s = \{x \mid \exists y \bullet\ y \in s \wedge x = tf\ n\ y\}$

HOL Constant

$\boldsymbol{CompoundFuns}$: $(\mathbb{N} \to 'a \to 'a) \to ('a\ SET \to 'a\ SET)\ LIST\ LIST$
$\quad \to ('a\ SET \to 'a\ SET)$

$\forall\ tf\ asfll \bullet\ CompoundFuns\ tf\ asfll =$
$\quad ListFunUnion\ (Map\quad ListFunUnion\ (Map\ (MapTag\ (LiftTag\ tf))\ asfll))$

HOL Constant

$\boldsymbol{HCF}$: $(\mathbb{N} \to 'a \to 'a) \to ('a\ SET \to 'a\ SET)\ LIST\ LIST \to 'a\ SET$

$\forall\ tf\ asfll \bullet\ HCF\ tf\ asfll = HeredFun\ (CompoundFuns\ tf\ asfll)$

HOL Constant

$\boldsymbol{LiftTag2}$: $(\mathbb{N} \to 'a \to 'a) \to (\mathbb{N} \to ('a \to 'a \to BOOL) \to ('a \to 'a \to BOOL))$

$\forall\ tf\ n\ r \bullet\ LiftTag2\ tf\ n\ r = \lambda x\ y \bullet \exists z \bullet\ r\ x\ z \wedge y = tf\ n\ z$

HOL Constant

$\boldsymbol{MapTag2}$: $(\mathbb{N} \to 'a \to 'a) \to ('a \to 'a \to BOOL)\ LIST \to ('a \to 'a \to BOOL)\ LIST$

$\forall\ tf\ rl \bullet\ MapTag2\ tf\ rl = MapNFun\ (\lambda n\ r\ x\ y \bullet \exists z \bullet\ y = tf\ n\ z \wedge r\ x\ z)\ rl$

HOL Constant

$\boldsymbol{ListRelUnion}$: $('a \to 'a \to BOOL)\ LIST \to ('a \to 'a \to BOOL)$

$\forall l\ x\ y \bullet\ ListRelUnion\ l\ x\ y \Leftrightarrow \exists r \bullet\ r \in_L l \wedge\ \ r\ x\ y$

HOL Constant

$\boldsymbol{CompoundRels}$: $(\mathbb{N} \to 'a \to 'a) \to ('a \to 'a \to BOOL)\ LIST\ LIST \to ('a \to 'a \to BOOL)$

$\forall\ tf\ arll \bullet\ CompoundRels\ tf\ arll =$
$\quad ListRelUnion\ (Map\quad ListRelUnion\ (Map\ (MapTag2\ tf)\ arll))$

HOL Constant

$\boldsymbol{HCR}$: $(\mathbb{N} \to 'a \to 'a) \to ('a \to 'a \to BOOL)\ LIST\ LIST \to 'a\ SET$

$\forall\ tf\ asrll \bullet\ HCR\ tf\ asrll = HeredRel\ (CompoundRels\ tf\ asrll)$

8 INDUCTIVE DEFINITIONS OF RELATIONS

This section is included to provided support for defining reduction systems over various kinds of terms. The two kinds of term with which I am most concerned are those of combinatory logic, and of infinitary combinatory logic. These are of course quite different, the material here is therefore independent of the type of terms involved.

This material is intended to provide an alternative to the facilities available in hol4 for defining relations. These facilities work from a notation for defining rules, and define the reflexive transitive closure of the given rules, delivering a couple of induction principles and some other useful theorems.

The approach here is to assume that primitive reductions are defined using the normal definitional features in HOL, and provide the obvious operators for obtaining the required kind of relation from these primitive reductions.

The two main operations of interest are that of forming the reflexive and transitive closures and that of making a congruence relation. The first two are independent of the term structure, but the last one is not. To reason about that operation independently of the structure we would need to work with parameters which expose some aspects of the structure.

9 PSEUDO (CO-)INDUCTIVE DEFINITIONS

This discussion is tainted by the attempt to support a tendentious application of the notion of pseudo-inductive definition to the construction of pure, non-well-founded ontologies of concrete categories and functors. I don't now have much inclination to believe that it will serve that purpose, but nevertheless would like to understand the notion of pseudo-inductiveness and get some idea of what it is good for. I therefore propose to remove from this section the material motivated by category theoretic ontology, and in the first instance focus on working over the ideas presented in [3].

Forster [3], introduces pseudo well-foundedness as way of getting a kind of inductive proof principle in NF. A pseudo-well-founded universe is partitioned into two clasess, in a way which suggests that the desired split of the universe into categories and functors might be definable by adaptation of this principle. To see the connection observe that the 'constituents' of categories are functors, and the constuents of functors are categories. This means that, in a well-founded universe, a slightly gerrimandered notion of rank (in which rank increases by exactly one as we pass from a functor to a category containing that functor or from a category to a functor over that category) assigns even ranks to categories and odd ranks to functors (we need to enhance the notion of odd and even to cover limit points, which I think need to be even). Not only are categories and functors distinguished by the parity of their rank, there is a difference in parity for every descending path according to whether it begins at a category or a functor. This fits in with the game theoretic flavour of pseudo well-foundedness in which a game is just such a descending path, and the winner of the game is determined by the parity of its length.

For these reasons I thought it might be interesting to see how a 'pseudo-inductive' definition adapted from Forster's notion of pseudo-well-foundedness worked in this application.

Since embarking on this path a second method occurred to me, which is to use co-induction. For the purposes which Forster had in mind (a principle which could be added to the axioms of NF to exclude certain possibly pathological sets (like a set which is its own unit set) while still leaving the important well-founded sets) co-induction would not suffice, since it looks like co-induction for what its worth, comes for free in NF. However, in this application, where we are filtering out all but categories and functors, the coinductive defintiion will actually do something, and the fact (if it

is a fact) that a principle of co-well-foundedness is derivable in NF would suggest that this form of definitions will give the most liberal ontology of the selected kind.

Once co-induction occurs as an alternative non-well-foundedness preserving definitional method, the question arised whether there is a similar dual to psuedo-well foundedness, and how this 'pseudo-co-induction' differs from pseduo- and co-induction, if at all.

9.1 Well Foundedness and Hereditarily Collections

It would be a good idea eventually to define these first and check back that they are special cases of the pseudo versions and that the well-founded functor/category collections are contained in the pseudo-well-founded collections.

9.2 Hereditarily and Co-Hereditarily PQ

Though [5] defines methods for inductive and co-inductive definition which generalise the notion of a 'hereditarily P' set for arbitrary properties of sets P, it does not explicitly cover the present requirement for mutually dependent inductive definitions. However, this can be achieved by two separate inductive definitions.

No additional machinery is required, the application to yield purely inductive and co-inductive conceptions of category and functor are shown in the next section.

9.3 Pseudo-Well-Founded Collections

I have previously done the kind of category theoretic construction in well-founded set theories. The same method could be used here, but the result would be similar, the constructed domains would still be well-founded.

I am therefore hoping to extract from Forster's discussion about pseudo-induction some idea about how to do the construction without throwing away the non-well-founded sets.

First we define the concept "pseudo-well-founded". The set theoretic language here (membership relation and power set) is the one which comes with ProofPower HOL (and is set theoretic syntax for predicates in a type theory), the target set theory is modelled as a membership relation (r) over some type (*X). This is based on the definition in Thomas Forster's book, and is included mainly for reference.

The presentation has been restructured. The restructuring serves three purposes. The first is to expose the two classes which partition the universe so that we can talk about them (in the original they are existentially quantified in the definition. This is done by splitting the definition into two parts, the existential quantification appearing in the second. The second is to include the claim that these two classes exhaust the type under consideration in the second part, so that the first definition can still be used in a set theory which is not known to be pseudo-well-founded for talking about its pseudo-well-founded part. Otherwise we would have to add an axiom to NFU for the construction we have in mind. The third change is to split our definitions of *I_closed* and *II_closed*, to make the structure of proofs more transparent.

HOL Constant

$$\boldsymbol{I\_closed} : ('X \to 'X \to BOOL) \to 'X\ \mathbb{P} \to BOOL$$

$$\forall r\ w \bullet\ I\_closed\ r\ w \\ \quad \Leftrightarrow \forall x \bullet\ (\exists y \bullet\ r\ y\ x \wedge \forall z \bullet\ r\ z\ y \Rightarrow z \in w) \Rightarrow x \in w$$

HOL Constant

$$\boldsymbol{II\_closed} : ('X \to 'X \to BOOL) \to 'X\ \mathbb{P} \to BOOL$$

$$\forall r\ w \bullet\ II\_closed\ r\ w \\ \quad \Leftrightarrow \forall x \bullet\ (\forall y \bullet\ r\ y\ x \Rightarrow \exists z \bullet\ r\ z\ y \wedge z \in w) \Rightarrow x \in w$$

HOL Constant

$$\boldsymbol{I\_closure} : ('X \to 'X \to BOOL) \to 'X\ \mathbb{P}$$

$$\forall r \bullet\ I\_closure\ r = \bigcap \{x \mid I\_closed\ r\ x\}$$

HOL Constant

$$\boldsymbol{II\_closure} : ('X \to 'X \to BOOL) \to 'X\ \mathbb{P}$$

$$\forall r \bullet\ II\_closure\ r = \bigcap \{x \mid II\_closed\ r\ x\}$$

HOL Constant

$$\boldsymbol{PseudoWellFounded} : ('X \to 'X \to BOOL) \to BOOL$$

$$\forall r \bullet\ PseudoWellFounded\ r \Leftrightarrow \\ \quad (I\_closure\ r) \cup (II\_closure\ r) = Universe$$

The following theorems correspond to the induction principles in [3], p14.

$$\boldsymbol{p\_induct\_thm1} = \\ \quad \vdash \forall\ X'\ R \\ \quad\quad \bullet\ (\forall\ x \bullet\ (\exists\ y \bullet\ R\ y\ x \wedge (\forall\ z \bullet\ R\ z\ y \Rightarrow z \in X')) \Rightarrow x \in X') \\ \quad\quad\quad \Rightarrow I\_closure\ R \subseteq X'$$

$$\boldsymbol{p\_induct\_thm2} = \\ \quad \vdash \forall\ X'\ R \\ \quad\quad \bullet\ (\forall\ x \bullet\ (\forall\ y \bullet\ R\ y\ x \Rightarrow (\exists\ z \bullet\ R\ z\ y \wedge z \in X')) \Rightarrow x \in X') \\ \quad\quad\quad \Rightarrow II\_closure\ R \subseteq X'$$

$$\vdash \forall\ R\ y\bullet\ (\forall\ x\bullet\ x \in y \Rightarrow I\_closed\ R\ x) \Rightarrow I\_closed\ R\ (\bigcap\ y)$$

II_closed_⋂_lemma =

$$\vdash \forall\ R\ y\bullet\ (\forall\ x\bullet\ x \in y \Rightarrow II\_closed\ R\ x) \Rightarrow II\_closed\ R\ (\bigcap\ y)$$

I_closed_closure_lemma =

$$\vdash \forall\ R\bullet\ I\_closed\ R\ (I\_closure\ R)$$

II_closed_closure_lemma =

$$\vdash \forall\ R\bullet\ II\_closed\ R\ (II\_closure\ R)$$

The idea is to use this to define a notion of "pseudo-hereditarily P" set for a property of sets P, analogously to the notion of a "heridtarily P" set in a well-founded set theory.

This is a bit too simplistic, because the idea of hereditarily P sets is *thin* in the following sense. To prove that a set is hereditarily P you need only to know that each of its members is hereditarily P. So, in a sense you are only looking one level down. When you do this with categories and functors the categories and functors alternate as the rank increases. You either need mutually recursive definitions or else the independent definitions of categories and functors have to look down two levels to express the required condition. There is already something like this going on in the pseudo-well-foundedness condition. I've no idea at present whether these work together or not.

9.4 Pseudo-Well-Founded-PQ Collections

This is derived from pseudo-well-foundedness by the following generalisations:

- instead of looking down one level at a time according to the membership relation, the property is parameterised by a function which extracts the 'content' from an object. This might be just the members, or it might be, say, the field of a function.
- for each of the two classes in question there is an additional parameterised property which is required to be satisfied.

Note that we have apparently lost the membership relation here, which is effectively hidden in the content extractors c and d.

The structure of this definition is now broken down similarly to that of *PseudoWellFounded*.

HOL Constant

P_closed : $(('X \to 'X\ \mathbb{P}) \times ('X \to 'X\ \mathbb{P}) \times ('X \to BOOL) \times ('X \to BOOL))$
$\quad \to 'X\ \mathbb{P} \to BOOL$

$$\forall c\ d\ P\ Q\ w\bullet\ P\_closed\ (c,\ d,\ P,\ Q)\ w$$
$$\Leftrightarrow \forall x\bullet\ P\ x \wedge (\forall y\bullet\ y \in (c\ x) \Rightarrow Q\ y \wedge (\exists z\bullet\ z \in (d\ y) \wedge z \in w)) \Rightarrow x \in w$$

HOL Constant

$$\boldsymbol{Q\_closed} : ((\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to BOOL) \times (\prime X \to BOOL)) \to \prime X\ \mathbb{P} \to BOOL$$

$$\forall c\ d\ P\ Q\ w \bullet\ Q\_closed\ (c,\ d,\ P,\ Q)\ w \Leftrightarrow \forall x \bullet\ Q\ x \wedge (\exists y \bullet\ y \in (d\ x) \wedge P\ y \wedge \forall z \bullet\ z \in (c\ y) \Rightarrow z \in w) \Rightarrow x \in w$$

HOL Constant

$$\boldsymbol{P\_closure} : ((\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to BOOL) \times (\prime X \to BOOL)) \to \prime X\ \mathbb{P}$$

$$\forall z \bullet\ P\_closure\ z = \bigcap \{x \mid P\_closed\ z\ x\}$$

HOL Constant

$$\boldsymbol{Q\_closure} : ((\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to BOOL) \times (\prime X \to BOOL)) \to \prime X\ \mathbb{P}$$

$$\forall z \bullet\ Q\_closure\ z = \bigcap \{x \mid Q\_closed\ z\ x\}$$

HOL Constant

$$\boldsymbol{PseudoWellFoundedPQ} : ((\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to BOOL) \times (\prime X \to BOOL)) \to BOOL$$

$$\forall z \bullet\ PseudoWellFoundedPQ\ z \Leftrightarrow (P\_closure\ z) \cap (Q\_closure\ z) = \{\} \wedge (P\_closure\ z) \cup (Q\_closure\ z) = Universe$$

Need to check that this encompasses pseudo-well-foundedness when suitably parameterised and that the 'pseudo' bit only makes a difference for non-well-founded membership structures.

$$\boldsymbol{P\_closed\_\Rightarrow\_P\_thm} = \vdash \forall\ c\ d\ P\ Q\ x \bullet\ x \in P\_closure\ (c,\ d,\ P,\ Q) \Rightarrow P\ x$$

$$\boldsymbol{Q\_closed\_\Rightarrow\_Q\_thm} = \vdash \forall\ c\ d\ P\ Q\ x \bullet\ x \in Q\_closure\ (c,\ d,\ P,\ Q) \Rightarrow Q\ x$$

9.5 Pseudo-Hereditarily-PQ Collections

Even if we had a pseudo-well-founded universe in NFU, it would not be pseudo-well-founded-PQ for any interesting or useful properties P and Q. The whole idea is to use these properties to create a pair of useful subclasses of the universe. So what we want to do is just take the two closures resulting from P and Q, and use their union as new a psuedo-well-founded domain of discourse. In fact, we won't take the union at all, we will be looking for a two-sorted foundational theory.

HOL Constant

$$\textbf{\textit{PseudoHereditarilyPQ}} : ((\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to \prime X\ \mathbb{P}) \times (\prime X \to BOOL) \times (\prime X \to BOOL)) \to (\prime X\ \mathbb{P} \times \prime X\ \mathbb{P})$$

$$\forall z \bullet PseudoHereditarilyPQ\ z = (P\_closure\ z,\ Q\_closure\ z)$$

Among the important properties which we hope this construction will have are:

- The values in the left collection will all have property P.
- The values in the right collection will all have property Q.
- Both collections will include the corresponding *HereditarilyPQ* sets.

I am by no means confident of any of these features at present.

10 A RECURSION PRINCIPLE

An induction principle is often associated with a notion of well-foundedness such that some collection will be well-founded iff the induction principle holds over the collection. There will then be a recursion theorem which tells us about the existence of functions satisfying recursive equations, to the effect that if the functor which captures the content of the recursion equations respects the well-founded relation then it will have a fixed point which is a solution to the equations.

In this pattern there is usually a clean separation between the well-founded relation and the functor which respects that relation, and to prove the existence of the fixed point using the recursion theorem one must show that the relation is well-founded (which may be done without reference to the functor) and then show that the functor respects the relation (which will not usually depend on its being well-founded).

The idea of a functor respecting some relation captures the idea that in evaluating the recursive function each time a recursion takes place the argument of the function is lower in some well-founded relation than the previous invocation, which guarantees that the recursion must eventually terminate. The well founded relation therefore must capture the relationship "in evaluating f at y reference may be made to its value at x".

This relationship being simply a relationship over the domain of the proposed function, it cannot take into account the values of f, but a recursive definition of a function clearly can do so. A function whose evaluation terminates only because its dependencies are sensitive to values of the function may not be provably terminating by these means.

Elsewhere I address the problem of constructing non-well-founded interpretations of set theory by seeking fixed points of recursive definitions of membership over suitable domains. In some cases these constructions may be connected with problems which have proven very hard to solve, for example the consistency of Quine's NF, so it is reasonable to expect that proving the existence of a fixed point in such a case may depend upon a recursion principle stronger than the usual kind. This is what I am attempting to fcrmulate.

To formulate the principle some auxiliary notions are helpful. I need to ba able to talk about a region over which the values of a function are fixed under the operation of a functor.

HOL Constant

$$\boldsymbol{FunEquiv}: ('a \to 'b) \to ('a \to 'b) \to 'a\ SET \to BOOL$$

$$\forall f\ g\ X \bullet FunEquiv\ f\ g\ X \Leftrightarrow \forall x \bullet x \in X \Rightarrow g\ x = f\ x$$

HOL Constant

$$\boldsymbol{FixedRegion}: (('a \to 'b) \to ('a \to 'b)) \to ('a \to 'b) \to 'a\ SET \to BOOL$$

$$\forall G\ f\ X \bullet FixedRegion\ G\ f\ X \Leftrightarrow$$
$$\forall g \bullet (\forall x \bullet x \in X \Rightarrow g\ x = f\ x) \Rightarrow (\forall x \bullet x \in X \Rightarrow G\ g\ x = f\ x)$$

What I aim to say here is that if a functor satisfies something a bit like and induction principle then it has a fixed point.

$$\boldsymbol{special\_recursion\_thm} =$$
$$\vdash ?\ \forall\ G \bullet$$
$$(\forall P\ x \bullet (\exists v\ X\ f \bullet \forall g \bullet (\forall y \bullet y \in X \Rightarrow g\ y = f\ y)$$
$$\Rightarrow G\ g\ x = v \wedge (\forall y \bullet y \in X \Rightarrow P\ y)) \Rightarrow P\ x)$$
$$\Rightarrow \exists g \bullet G\ g = g$$

11 CODING CONSTRUCTIONS

It is proposed in the first instance to use HOL types as ways of describing constructions.

It is also desirable to allow the domains of the constructors to be specified by predicates, especially since some type constructors like the power set will not be implementable without some constraint on the subsets.

11.1 Constructor Translation Kits

In order to make the translation of such a description of a system of constructors into a function whose fixedpoint can be taken to yield an implementation, the code will be parameterised by the necessary things to build the function. For example, for type constructor allowed in the description a function is required to perform the construction of elements of that type over the chosen representation type. So we need a map (in SML) from HOL type constructors to hol functions containing this information.

The type CTK is defined as a 1-ary type constructor which in its in its primary role instantiated to type $TERM$. When instantiated to *string* the type can be used to package a set of aliases for these terms enabling the complex terms constructed for fixed points to be presented concisely in theory listings. The aliases may be used temporarily just for the theory listing and then easily undeclared so that they do not interfere with parsing.

SML

$type\ 'a\ \boldsymbol{CTK} = \{$
$\quad tag : 'a,$
$\quad tagc : 'a,$
$\quad mk\_leaf: 'a,$
$\quad node\_constructors: 'a\ list,$
$\quad content\_extractors: 'a\ list,$
$\quad leaf\_injections: 'a\ list,$
$\quad leaf\_content: 'a,$
$\quad ccp\_converter: 'a,$
$\quad compound\_fixp: 'a\};$

11.2 Lifting Over HOL Type Constructors

The following functions allow the coding of constructors over a representation type to be written without concern for how the functions will be compounded. They lift an operator over a representation type to be lifted to a functor which transforms constructors for the operand types into a constructor for the result type.

These lifting functions are specific to the HOL type constructors but independent of the representation type.

HOL Constant

$\boldsymbol{LiftProduct}: ('t \times 't \to 't) \to ('a \to 't) \to ('b \to 't) \to ('a \times 'b \to 't)$

$\forall prod\ f\ g\ a\ b \bullet LiftProduct\ prod\ f\ g\ (a,\ b) = prod\ (f\ a,\ g\ b)$

HOL Constant

$\boldsymbol{LiftSum}: ('t + 't \to 't) \to ('a \to 't) \to ('b \to 't) \to ('a + 'b \to 't)$

$\forall sum\ f\ g\ ab \bullet LiftSum\ sum\ f\ g\ ab = sum\ ((Fun_+\ f\ g)\ ab)$

HOL Constant

$\boldsymbol{LiftList}: ('t\ LIST \to 't) \to ('a \to 't) \to ('a\ LIST \to 't)$

$\forall list\ f\ al \bullet LiftList\ list\ f\ al = list\ (Map\ f\ al)$

HOL Constant

$\boldsymbol{LiftSumUnion}: ('a \to 'u\ SET) \to ('b \to 'u\ SET) \to ('a + 'b \to 'u\ SET)$

$\forall cfl\ cfr\ ab \bullet LiftSumUnion\ cfl\ cfr\ ab = if\ IsL\ ab\ then\ cfl\ (OutL\ ab)\ else\ cfr\ (OutR\ ab)$

HOL Constant

$\boldsymbol{LiftSumPred}: ('a \to BOOL) \to ('b \to BOOL) \to ('a + 'b \to BOOL)$

$\forall pl\ pr\ ab \bullet LiftSumPred\ pl\ pr\ ab = if\ IsL\ ab\ then\ pl\ (OutL\ ab)\ else\ pr\ (OutR\ ab)$

11.3 Constructors Over Trees

Definitions of constructors for products and lists, and injections for disjoint unions.

I'm trying to make this a general as possible, the idea is to allow for the construction of inductive datatypes based on arbitrary trees.

Then constructions may are therefore be thought of as parameterised by:

1. a type of leaf values
2. a type of arc tags

And the tree we construct is a tree whose paths may be as long as the well-ordering and whose nodes and arcs have the appropriate kinds of tags.

A tree is then "implemented" as a function from paths to node tags, where a path is itself a function from an initial segment of the domain of the well-ordering into the arc tags. Both of these kinds of function are partial and are therefore represented as relations.

The fact that we are dealing with sets and partial function is here immaterial to the definition of the constructor function, and can be taken account of later in the process of constructing the inductive datatype, so the parameterisation is evident at this point only by the presence of three type variables.

I can't see how to make use of a well-ordering larger than the natural numbers so that's fixed, and that makes a path into a list of arcs.

11.3.1 Some Types, Some Properties

To make the type information in the theory listing less cluttered I will use some labelled product definitions to introduce types.

HOL Labelled Product

TREEE

$$\textbf{\textit{Treee}}: {}'arc\ LIST \to {}'leaf \to BOOL$$

One could make a proper subtype here by incorporating the necessary conditions on the relations involved, but in the construction of an inductive datatype there will almost invariably be another subtyping involved and one will probably suffice. However, it may be helpful at this point to define the conditions for a "TREEE" to be a tree. The conditions depend upon chosing a well-ordering of the index types, and are therefore parameterised by such a well-ordering. There may not be to allow for this to be a well-ordering of a subset of the type, but since the definition of well-ordering we have is defined in such terms (i.e. as a property of set/relation pairs) we will define the well-formedness condition as accepting such a well-ordering.

HOL Constant

$$\textbf{\textit{Domain}} : ({}'a \to {}'b \to BOOL) \to {}'a\ SET$$

$$\forall r \bullet\ Domain\ r = \{x \mid \exists y \bullet\ r\ x\ y\}$$

HOL Constant

$$\boldsymbol{NullTreee} : (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE}$$

$$\mathit{NullTreee} = \mathit{MkTREEE}(\lambda \mathit{al}\ \mathit{n}\bullet\ \mathit{F})$$

HOL Constant

$$\boldsymbol{IsTreee} : (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE} \to \mathit{BOOL}$$

$$\begin{array}{l}
\forall\ \mathit{t}{:}(\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE}\bullet \\
\quad \mathit{IsTreee}\ \mathit{t} \Leftrightarrow \\
\quad \neg\ \mathit{t} = \mathit{NullTreee} \\
\quad \wedge\ \mathit{ManyOne}\ (\mathit{Treee}\ \mathit{t}) \\
\quad \wedge\ (\forall \mathit{p}\ \mathit{q}\bullet\ \mathit{p} \in \mathit{Domain}\ (\mathit{Treee}\ \mathit{t}) \Rightarrow \neg\ \mathit{Append}\ \mathit{p}\ \mathit{q} \in \mathit{Domain}\ (\mathit{Treee}\ \mathit{t}))
\end{array}$$

11.3.2 A Generic Constructor

We have a single constructor function which takes a 'arc indexed set of trees and a 'leaf value. This constructs a new tree from whose root node has the supplied value and whose children are the supplied trees, placed on the arc names used to index them.

The paths through the tree are therefore the paths of the original trees with a new arc slotted in at the head of each path.

HOL Constant

$$\boldsymbol{MkTreee} : (\mathit{'arc} \to (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE} \to \mathit{BOOL}) \to (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE}$$

$$\begin{array}{l}
\forall \mathit{c}\bullet\ \mathit{MkTreee}\ \mathit{c} = \mathit{MkTREEE}\ (\lambda\ \mathit{path}\ \mathit{leaf}\bullet \\
\quad \exists\ \mathit{tr}\bullet\ \mathit{c}\ (\mathit{Hd}\ \mathit{path})\ \mathit{tr} \wedge \mathit{Treee}\ \mathit{tr}\ (\mathit{Tail}\ \mathit{path})\ \mathit{leaf})
\end{array}$$

Though this was written for use in inductive datatypes, I don't know a reason why it could not be used for co-inductive datatypes. Presumably you have the same kind of definition but take the maximal rather than the minimal fixed point.

This construction is only one-one subject to some constraints on the supplied map, as follows:

HOL Constant

$$\boldsymbol{NiceChildren} : (\mathit{'arc} \to (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE} \to \mathit{BOOL}) \to \mathit{BOOL}$$

$$\forall \mathit{c}\bullet\ \mathit{NiceChildren}\ \mathit{c} \Leftrightarrow \mathit{ManyOne}\ \mathit{c} \wedge \neg\ \exists \mathit{a}\bullet\ \mathit{c}\ \mathit{a}\ \mathit{NullTreee}$$

HOL Constant

$$\boldsymbol{DestTreee} : (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE} \to (\mathit{'arc} \to (\mathit{'arc}, \mathit{'leaf})\,\mathit{TREEE} \to \mathit{BOOL})$$

$$\forall \mathit{c}\bullet\ \mathit{NiceChildren}\ \mathit{c} \Rightarrow \mathit{DestTreee}\ (\mathit{MkTreee}\ \mathit{c}) = \mathit{c}$$

11.3.3 Specific Constructors

The idea here is to code up the details of how to construct appropriate data for MkTreee for the commonly expected type constructors in an inductive datatype definition.

HOL Constant

$$\boldsymbol{MkLeafTreee} : 'leaf \to ('arc, 'leaf)\,TREEE$$

$$\forall l \bullet\ MkLeafTreee\ l = MkTREEE\ (\lambda arc\ leaf \bullet\ arc = [] \wedge leaf = l)$$

HOL Constant

$$\boldsymbol{MkProdTreee} : (BOOL \to 'arc) \to (('arc, 'leaf)\,TREEE) \times (('arc, 'leaf)\,TREEE) \to ('arc, 'leaf)\,TREEE$$

$$\forall ai\ l\ r \bullet\ MkProdTreee\ ai\ (l, r) = MkTreee\ (\lambda arc\ tree \bullet\ arc = (ai\ T) \wedge tree = l \vee arc = (ai\ F) \wedge tree = r)$$

HOL Constant

$$\boldsymbol{MkSumTreee} : (BOOL \to 'arc) \to (('arc, 'leaf)\,TREEE) + (('arc, 'leaf)\,TREEE) \to ('arc, 'leaf)\,TREEE$$

$$\forall ai\ t \bullet\ MkSumTreee\ ai\ t = MkTreee\ (\lambda arc\ tree \bullet\ if\ IsL\ t\ then\ arc = (ai\ T) \wedge tree = OutL\ t\ else\ arc = (ai\ F) \wedge tree = OutR\ t)$$

HOL Constant

$$\boldsymbol{MkArcTreee} : 'arc \to ('arc, 'leaf)\,TREEE \to ('arc, 'leaf)\,TREEE$$

$$\forall a\ t \bullet\ MkArcTreee\ a\ t = MkTreee\ (\lambda arc\ tree \bullet\ arc = a \wedge tree = t)$$

HOL Constant

$$\boldsymbol{MkTagTreee} : (\mathbb{N} \to 'arc) \to \mathbb{N} \to ('arc, 'leaf)\,TREEE \to ('arc, 'leaf)\,TREEE$$

$$\forall ai\ n\ t \bullet\ MkTagTreee\ ai\ n\ t = MkArcTreee\ (ai\ n)\ t$$

HOL Constant

$$\boldsymbol{IsTagTreee} : (\mathbb{N} \to 'arc) \to \mathbb{N} \to ('arc, 'leaf)\,TREEE \to BOOL$$

$$\forall ai\ n\ t \bullet\ IsTagTreee\ ai\ n\ t \Leftrightarrow \exists t2 \bullet\ t = MkTagTreee\ ai\ n\ t2$$

HOL Constant

$$\boldsymbol{UnTagTreee} : (\mathit{'arc}, \mathit{'leaf})\mathit{TREEE} \to (\mathit{'arc}, \mathit{'leaf})\mathit{TREEE}$$

$$\forall t\bullet \exists\ t2\ \ a\bullet\ t = \mathit{MkArcTreee}\ a\ t2 \Rightarrow \mathit{UnTagTreee}\ t = t2$$

HOL Constant

$$\boldsymbol{MkListTreee} : (\mathbb{N} \to \mathit{'arc}) \to (\mathit{'arc}, \mathit{'leaf})\mathit{TREEE}\ \mathit{LIST} \to (\mathit{'arc}, \mathit{'leaf})\mathit{TREEE}$$

$$\forall ai\ trl\bullet$$
$$\mathit{MkListTreee}\ ai\ trl = \mathit{MkTreee}$$
$$(\lambda arc\ tree\bullet \exists n\bullet\ arc = (ai\ \ n) \wedge (n,\ tree) \in \mathit{ListRel}\ trl)$$

Node injections will be dynamically constructed.

Arc injections are fixed in any particular implementation of these inductive datatypes, determined by the range of type constructors to be supported. The arc type has to have a cardinality which is an upper bound of the cardinalities of those required by the type constructors. So, for the above set of constructors $\mathbb{N}$ suffices and the arc injectors are therefore defined as follows:

HOL Constant

$$\boldsymbol{AiOneToN} : \mathit{ONE} \to \mathbb{N};\ \ \mathit{AiBoolToN} : \mathit{BOOL} \to \mathbb{N};\ \mathit{AiNToN} : \mathbb{N} \to \mathbb{N}$$

$$(\forall one\bullet\ \mathit{AiOneToN}\ one = 0)$$
$$\wedge\ (\forall b\bullet\ \mathit{AiBoolToN}\ b = \mathit{if}\ b\ \mathit{then}\ 1\ \mathit{else}\ 0)$$
$$\wedge\ (\forall n\bullet\ \mathit{AiNToN}\ n = n)$$

11.4 An $\mathbb{N}$ Tree Constructor Translator Kit

In order to make the definitions of types no more prolix than need be it is desirable not to use expressions in making up constructor toolkits. So we here define constants for making a default toolkit based on trees with natural number arcs.

11.4.1 Special Constructors

The following definition instantiates the tree constructors to the injections into $\mathbb{N}$ and lifts them to give the functions required for a CTK.

HOL Constant

$$\boldsymbol{NTreeeTag}: \mathbb{N} \to (\mathit{'a}\ \ \to (\mathbb{N}, \mathit{'leaf})\mathit{TREEE}) \to (\mathit{'a}\ \ \to (\mathbb{N}, \mathit{'leaf})\mathit{TREEE})$$

$$\forall n\ f\bullet\ \mathit{NTreeeTag}\ n\ f = (\mathit{MkTagTreee}\ \mathit{AiNToN}\ n)\ o\ f$$

HOL Constant

$$\boldsymbol{NTreeeIsTag}: \mathbb{N} \to (\mathbb{N}, \mathit{'leaf})\mathit{TREEE} \to \mathit{BOOL}$$

$$\mathit{NTreeeIsTag} = \mathit{IsTagTreee}\ \mathit{AiNToN}$$

11.4.2 Node Constructors

HOL Constant

$$\mathbf{NTreeeMkList}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE) \to (\prime a\ LIST \to (\mathbb{N}, \prime leaf)\, TREEE)$$

$$NTreeeMkList = LiftList\ (MkListTreee\ AiNToN)$$

HOL Constant

$$\mathbf{NTreeeMkProd}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE) \to (\prime b \to (\mathbb{N}, \prime leaf)\, TREEE) \to (\prime a \times \prime b \to (\mathbb{N}, \prime leaf)\, TREEE);$$
$$\mathbf{NTreeeMkSum}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE) \to (\prime b \to (\mathbb{N}, \prime leaf)\, TREEE) \to (\prime a + \prime b \to (\mathbb{N}, \prime leaf)\, TREEE)$$

$$NTreeeMkProd = LiftProduct\ (MkProdTreee\ AiBoolToN)$$
$$\wedge\ NTreeeMkSum = LiftSum\ (MkSumTreee\ AiBoolToN)$$

11.4.3 Content Extractors

HOL Constant

$$\mathbf{NTrListC}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE\ SET) \to (\prime a\ LIST \to (\mathbb{N}, \prime leaf)\, TREEE\ SET)$$

$$NTrListC = LiftList\ (\lambda sl\bullet \bigcup \{x \mid x \in_L sl\})$$

HOL Constant

$$\mathbf{NTrProdC}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE\ SET) \to (\prime b \to (\mathbb{N}, \prime leaf)\, TREEE\ SET) \to (\prime a \times \prime b \to (\mathbb{N}, \prime leaf)\, TREEE\ SET);$$
$$\mathbf{NTrSumC}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE\ SET) \to (\prime b \to (\mathbb{N}, \prime leaf)\, TREEE\ SET) \to (\prime a + \prime b \to (\mathbb{N}, \prime leaf)\, TREEE\ SET)$$

$$NTrProdC = LiftProduct\ (Uncurry\ \$\cup)$$
$$\wedge\ NTrSumC = LiftSum\ (\lambda x\bullet\ if\ IsL\ x\ then\ OutL\ x\ else\ OutR\ x)$$

HOL Constant

$$\mathbf{NTrLeafC}: (\prime a \to (\mathbb{N}, \prime leaf)\, TREEE\ SET)$$

$$\forall l\bullet\ NTrLeafC\ l = \{\}$$

This function tags content as required by content extractors on type variables for the new types. It also makes a unit set of the result.

HOL Constant

$$\mathbf{NTreeeTagC}: \mathbb{N} \to (\mathbb{N}, \prime leaf)\, TREEE \to (\mathbb{N}, \prime leaf)\, TREEE\ SET$$

$$\forall n\ t\bullet\ NTreeeTagC\ n\ t = \{MkTagTreee\ AiNToN\ n\ t\}$$

11.4.4 Ntree Ctk

The following definition gives the required ccp to relation conversion:

HOL Constant

$$\boldsymbol{CR}: (\prime b \to \prime a) \to (\prime b \to \prime a\ SET) \to (\prime b \to BOOL) \to (\prime a \to \prime a \to BOOL)$$

$$\forall tor\ tent\ pred \bullet\ CR\ tor\ tent\ pred = CCP2Rel\ (MkCCP\ tor\ tent\ pred)$$

This definition provides the compounder which combines the relations for all the constructors and takes the fixed point.

HOL Constant

$$\boldsymbol{FR}: ((\mathbb{N}, \prime a)\ TREEE \to (\mathbb{N}, \prime a)\ TREEE \to BOOL)\ LIST\ LIST \to (\mathbb{N}, \prime a)\ TREEE\ SET$$

$$\forall rll \bullet\ FR\ rll = HeredRel\ (CompoundRels\ (MkTagTreee\ AiNToN)\ rll)$$

SML

```
val ntree_ctk:TERM CTK = {
    tag = ⌜NTreeeTag⌝,
    tagc = ⌜NTreeeTagC⌝,
    mk_leaf = ⌜MkLeafTreee⌝,
    node_constructors = [⌜NTreeeMkProd⌝, ⌜NTreeeMkSum⌝, ⌜NTreeeMkList⌝],
    leaf_injections = [⌜MkLeafTreee:CHAR→(ℕ, CHAR)TREEE⌝],
    content_extractors = [⌜NTrProdC⌝, ⌜NTrSumC⌝, ⌜NTrListC⌝],
    leaf_content = ⌜NTrLeafC⌝,
    ccp_converter = ⌜CR⌝,
    compound_fixp = ⌜FR⌝
};
val ctk_aliases:string CTK ={
    tag = "υ",
    tagc = "ω",
    mk_leaf = "γ",
    node_constructors = ["×", "+", "ϕ"],
    leaf_injections = ["ρ"],
    content_extractors = ["×", "+", "ϕ"],
    leaf_content = "ρ",
    ccp_converter = "Ξ",
    compound_fixp = "Θ"
};

fun declare_ctk_aliases (ctk: TERM CTK) (sctk:string CTK) =
 let
    fun map_declare_alias (sl, tl) = map declare_alias (combine sl tl);
    val _ = declare_alias (#tag sctk, #tag ctk);
    val _ = declare_alias (#tagc sctk, #tagc ctk);
    val _ = declare_alias (#mk_leaf sctk, #mk_leaf ctk);
```

```
    val _ = map_declare_alias (#node_constructors sctk, #node_constructors ctk);
    val _ = map_declare_alias (#leaf_injections sctk, #leaf_injections ctk);
    val _ = map_declare_alias (#content_extractors sctk, #content_extractors ctk);
    val _ = declare_alias (#leaf_content sctk, #leaf_content ctk);
    val _ = declare_alias (#ccp_converter sctk, #ccp_converter ctk);
    val _ = declare_alias (#compound_fixp sctk, #compound_fixp ctk)
    in ()
    end;

fun undeclare_ctk_aliases (ctk: TERM CTK) (sctk:string CTK) =
 let
    fun map_undeclare_alias (sl, tl) = map undeclare_alias (combine sl tl);
    val _ = undeclare_alias (#tag sctk, #tag ctk);
    val _ = undeclare_alias (#tagc sctk, #tagc ctk);
    val _ = undeclare_alias (#mk_leaf sctk, #mk_leaf ctk);
    val _ = map_undeclare_alias (#node_constructors sctk, #node_constructors ctk);
    val _ = map_undeclare_alias (#leaf_injections sctk, #leaf_injections ctk);
    val _ = map_undeclare_alias (#content_extractors sctk, #content_extractors ctk);
    val _ = undeclare_alias (#leaf_content sctk, #leaf_content ctk);
    val _ = undeclare_alias (#ccp_converter sctk, #ccp_converter ctk);
    val _ = undeclare_alias (#compound_fixp sctk, #compound_fixp ctk)
    in ()
    end;
```

12 MAKING NEW TYPES

The last stage in this process takes place where the objective is to introduce new types and constructors over the new types. The preceding material enables the description of a system of constructors to be translated into a realisation of that system over various sets.

This final stage involves the creation of a new type (possibly a type constructor) for each of the distinct sets which are the co-domains of the constructors, the definition of operators over these types corresponding to the constructors, and the transfer of the properties of the sets to theorems over the types.

These facilities will I hope be orthogonal to most aspects of variation in the definition of the underlying sets. It is most strongly motivated by the recursive datatypes of computer science, where there is a desire to abstract away from the details of how the constructions are coded, and may not be required in metatheoretic applications. An example of the latter applications is where the system of constructors is providing a coding of syntax into some domain of interest (as in Goedel numbering). However, in other metatheoretic studies, where the details of the construction are important and specific to the application, e.g. in the definition of mutually dependent hereditarily pure concrete functors and categories, it may be desired at the end to abstract away from the constructions and have separate new types for functors and categories.

13 Proof Contexts

SML

```
commit_pc "'fixp";

force_new_pc "fixp";
merge_pcs ["rbjmisc", "'fixp"] "fixp";
commit_pc "fixp";

force_new_pc "fixp1";
merge_pcs ["rbjmisc1", "'fixp"] "fixp1";
commit_pc "fixp1";
```

14 The Theory fixp

14.1 Parents

wf_relp *U_orders* *rbjmisc*

14.2 Children

misc1

14.3 Constants

IsLb	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a \to BOOL$
IsUb	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a \to BOOL$
IsGlb	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a \to BOOL$
IsLub	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a \to BOOL$
Lub	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a$
Glb	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to 'a$
NeGlbsExist	$('a \to 'a \to BOOL) \to BOOL$
GlbsExist	$('a \to 'a \to BOOL) \to BOOL$
LubsExist	$('a \to 'a \to BOOL) \to BOOL$
Monotonic	$('a\ \mathbb{P} \to 'b\ \mathbb{P}) \to BOOL$
$***ClosedUnder***	$'a\ \mathbb{P} \to ('a\ \mathbb{P} \to 'a\ \mathbb{P}) \to BOOL$
$***OpenUnder***	$'a\ \mathbb{P} \to ('a\ \mathbb{P} \to 'a\ \mathbb{P}) \to BOOL$
IsLfp	$('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a \to BOOL$
IsGfp	$('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a \to BOOL$
Lfp	$('a\ \mathbb{P} \to 'a\ \mathbb{P}) \to 'a\ \mathbb{P}$
Gfp	$('a\ \mathbb{P} \to 'a\ \mathbb{P}) \to 'a\ \mathbb{P}$
Rpo	$'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
RpoU	$('a \to 'a \to BOOL) \to BOOL$
$***Lfp_c***$	$('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a$
$***Gfp_c***$	$('a \to 'a \to BOOL) \to ('a \to 'a) \to 'a$
Directed	$('a \to 'a \to BOOL) \to 'a\ \mathbb{P} \to BOOL$
DirectedUb	$('a \to 'a \to BOOL) \to BOOL$
Increasing	$('a \to 'a \to BOOL) \to ('b \to 'b \to BOOL) \to ('a \to 'b) \to BOOL$
ChainComplete	$'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
CcRpo	$'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
CcRpoU	$('a \to 'a \to BOOL) \to BOOL$
CRpo	$('a \to 'a \to BOOL) \to BOOL$
FClosed	$('a \to 'a) \to 'a\ \mathbb{P} \to BOOL$
FChainClosed	$('a \to 'a) \to 'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
FChain	$('a \to 'a) \to 'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to 'a\ \mathbb{P}$
FChainU	$('a \to 'a) \to ('a \to 'a \to BOOL) \to 'a\ \mathbb{P}$
Extreme	$('a \to 'a) \times 'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to 'a\ \mathbb{P}$
S	$('a \to 'a) \times 'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to 'a \to 'a\ \mathbb{P}$
RelInv	$('a \to 'a \to BOOL) \to 'a \to 'a \to BOOL$
ChainCoComplete	$'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
CoCcRpo	$'a\ \mathbb{P} \times ('a \to 'a \to BOOL) \to BOOL$
CoCcRpoU	$('a \to 'a \to BOOL) \to BOOL$

$\boldsymbol{FCoChainClosed}$
$(\prime a \to \prime a) \to \prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to BOOL$

$\boldsymbol{FCoChain}$ $(\prime a \to \prime a) \to \prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P}$

$\boldsymbol{Closed_c}$ $(\prime a \to \prime a \to BOOL) \times (\prime a \to \prime a) \to \prime a \to BOOL$

$\boldsymbol{Closure_c}$ $(\prime a \to \prime a \to BOOL) \times (\prime a \to \prime a) \to \prime a \to \prime a$

$\boldsymbol{Closure_d}$ $(\prime a \to \prime a \to BOOL) \times (\prime a \to \prime a) \to \prime a \to \prime a$

$\boldsymbol{CRpoU}$ $(\prime a \to \prime a \to BOOL) \to BOOL$

$\boldsymbol{ContProp}$ $(\prime a \to \prime a \to BOOL) \to (\prime a \to BOOL) \to BOOL$

$\boldsymbol{Rising}$ $(\prime a \to \prime a) \to (\prime a \to \prime a \to BOOL) \to \prime a \to BOOL$

$\boldsymbol{Falling}$ $(\prime a \to \prime a) \to (\prime a \to \prime a \to BOOL) \to \prime a \to BOOL$

$\boldsymbol{ContProp3}$ $(\prime a \to \prime a) \to (\prime a \to \prime a \to BOOL) \to (\prime a \to BOOL) \to BOOL$

$\boldsymbol{Fun2MonoFun}$ $(\prime a\ \mathbb{P} \to \prime a\ \mathbb{P}) \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{HeredFun}$ $(\prime a\ \mathbb{P} \to \prime a\ \mathbb{P}) \to \prime a\ \mathbb{P}$

$\boldsymbol{CoHeredFun}$ $(\prime a\ \mathbb{P} \to \prime a\ \mathbb{P}) \to \prime a\ \mathbb{P}$

$\$\boldsymbol{ClosedUnderFun}$
$\prime a\ \mathbb{P} \to (\prime a\ \mathbb{P} \to \prime a\ \mathbb{P}) \to BOOL$

$\$\boldsymbol{OpenUnderFun}$
$\prime a\ \mathbb{P} \to (\prime a\ \mathbb{P} \to \prime a\ \mathbb{P}) \to BOOL$

$\boldsymbol{Rules2Fun}$ $\prime a\ \mathbb{P} \leftrightarrow \prime a \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{Rel2Fun}$ $(\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{HeredRel}$ $(\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P}$

$\boldsymbol{CoHeredRel}$ $(\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P}$

$\$\boldsymbol{ClosedUnderRel}$
$\prime a\ \mathbb{P} \to (\prime a \to \prime a \to BOOL) \to BOOL$

$\$\boldsymbol{OpenUnderRel}$
$\prime a\ \mathbb{P} \to (\prime a \to \prime a \to BOOL) \to BOOL$

$\boldsymbol{Prop2Rel}$ $\prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to (\prime a \to BOOL) \to \prime a \to \prime a \to BOOL$

$\boldsymbol{Hereditary}$ $\prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to (\prime a \to BOOL) \to \prime a\ \mathbb{P}$

$\boldsymbol{CoHereditary}$ $\prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to (\prime a \to BOOL) \to \prime a\ \mathbb{P}$

$\boldsymbol{ClosedUnderProp}$
$\prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P} \to (\prime a \to BOOL) \to BOOL$

$\boldsymbol{OpenUnderProp}$
$\prime a\ \mathbb{P} \times (\prime a \to \prime a \to BOOL) \to \prime a\ \mathbb{P} \to (\prime a \to BOOL) \to BOOL$

$\boldsymbol{Predicate}$ $(\prime a, \prime b)\ CCP \to \prime b \to BOOL$

$\boldsymbol{Content}$ $(\prime a, \prime b)\ CCP \to \prime b \to \prime a\ \mathbb{P}$

$\boldsymbol{Constructor}$ $(\prime a, \prime b)\ CCP \to \prime b \to \prime a$

$\boldsymbol{MkCCP}$ $(\prime b \to \prime a) \to (\prime b \to \prime a\ \mathbb{P}) \to (\prime b \to BOOL) \to (\prime a, \prime b)\ CCP$

$\boldsymbol{CCP2Fun}$ $(\prime a, \prime b)\ CCP \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{CCP2Rel}$ $(\prime a, \prime b)\ CCP \to \prime a \to \prime a \to BOOL$

$\boldsymbol{MapNFun}$ $(\mathbb{N} \to \prime a \to \prime a) \to \prime a\ LIST \to \prime a\ LIST$

$\boldsymbol{MapTag}$ $(\mathbb{N} \to \prime a \to \prime a) \to (\prime a \to \prime a)\ LIST \to (\prime a \to \prime a)\ LIST$

$\boldsymbol{LiftTag}$ $(\mathbb{N} \to \prime a \to \prime a) \to \mathbb{N} \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{CompoundFuns}$ $(\mathbb{N} \to \prime a \to \prime a) \to (\prime a\ \mathbb{P} \to \prime a\ \mathbb{P})\ LIST\ LIST \to \prime a\ \mathbb{P} \to \prime a\ \mathbb{P}$

$\boldsymbol{HCF}$ $(\mathbb{N} \to \prime a \to \prime a) \to (\prime a\ \mathbb{P} \to \prime a\ \mathbb{P})\ LIST\ LIST \to \prime a\ \mathbb{P}$

$\boldsymbol{LiftTag2}$ $(\mathbb{N} \to \prime a \to \prime a) \to \mathbb{N} \to (\prime a \to \prime a \to BOOL) \to \prime a \to \prime a \to BOOL$

$\boldsymbol{MapTag2}$ $(\mathbb{N} \to \prime a \to \prime a)$
$\to (\prime a \to \prime a \to BOOL)\ LIST$
$\to (\prime a \to \prime a \to BOOL)\ LIST$

$\boldsymbol{ListRelUnion}$ $(\prime a \to \prime a \to BOOL)\ LIST \to \prime a \to \prime a \to BOOL$

$\boldsymbol{CompoundRels}$ $(\mathbb{N} \to \prime a \to \prime a)$
$\to (\prime a \to \prime a \to BOOL)\ LIST\ LIST$

$\to {'a}$
$\to {'a}$
$\to BOOL$

HCR $(\mathbb{N} \to {'a} \to {'a}) \to ({'a} \to {'a} \to BOOL)\ LIST\ LIST \to {'a}\ \mathbb{P}$

HereditarilyPQ
$({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P} \times {'X}\ \mathbb{P}$

I_closed $({'X} \to {'X} \to BOOL) \to {'X}\ \mathbb{P} \to BOOL$

II_closed $({'X} \to {'X} \to BOOL) \to {'X}\ \mathbb{P} \to BOOL$

I_closure $({'X} \to {'X} \to BOOL) \to {'X}\ \mathbb{P}$

II_closure $({'X} \to {'X} \to BOOL) \to {'X}\ \mathbb{P}$

PseudoWellFounded
$({'X} \to {'X} \to BOOL) \to BOOL$

P_closed $({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P}$
$\to BOOL$

Q_closed $({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P}$
$\to BOOL$

P_closure $({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P}$

Q_closure $({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P}$

PseudoWellFoundedPQ
$({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to BOOL$

PseudoHereditarilyPQ
$({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to {'X}\ \mathbb{P}) \times ({'X} \to BOOL) \times ({'X} \to BOOL)$
$\to {'X}\ \mathbb{P} \times {'X}\ \mathbb{P}$

FunEquiv $({'a} \to {'b}) \to ({'a} \to {'b}) \to {'a}\ \mathbb{P} \to BOOL$

FixedRegion $(({'a} \to {'b}) \to {'a} \to {'b}) \to ({'a} \to {'b}) \to {'a}\ \mathbb{P} \to BOOL$

LiftProduct $({'t} \times {'t} \to {'t}) \to ({'a} \to {'t}) \to ({'b} \to {'t}) \to {'a} \times {'b} \to {'t}$

LiftSum $({'t} + {'t} \to {'t}) \to ({'a} \to {'t}) \to ({'b} \to {'t}) \to {'a} + {'b} \to {'t}$

LiftList $({'t}\ LIST \to {'t}) \to ({'a} \to {'t}) \to {'a}\ LIST \to {'t}$

LiftSumUnion $({'a} \to {'u}\ \mathbb{P}) \to ({'b} \to {'u}\ \mathbb{P}) \to {'a} + {'b} \to {'u}\ \mathbb{P}$

LiftSumPred $({'a} \to BOOL) \to ({'b} \to BOOL) \to {'a} + {'b} \to BOOL$

Treee $({'arc}, {'leaf})\ TREEE \to {'arc}\ LIST \to {'leaf} \to BOOL$

MkTREEE $({'arc}\ LIST \to {'leaf} \to BOOL) \to ({'arc}, {'leaf})\ TREEE$

Domain $({'a} \to {'b} \to BOOL) \to {'a}\ \mathbb{P}$

NullTreee $({'arc}, {'leaf})\ TREEE$

IsTreee $({'arc}, {'leaf})\ TREEE \to BOOL$

MkTreee $({'arc} \to ({'arc}, {'leaf})\ TREEE \to BOOL)$
$\to ({'arc}, {'leaf})\ TREEE$

NiceChildren $({'arc} \to ({'arc}, {'leaf})\ TREEE \to BOOL) \to BOOL$

DestTreee $({'arc}, {'leaf})\ TREEE \to {'arc} \to ({'arc}, {'leaf})\ TREEE \to BOOL$

MkLeafTreee ${'leaf} \to ({'arc}, {'leaf})\ TREEE$

MkProdTreee $(BOOL \to {'arc})$
$\to ({'arc}, {'leaf})\ TREEE \times ({'arc}, {'leaf})\ TREEE$
$\to ({'arc}, {'leaf})\ TREEE$

MkSumTreee $(BOOL \to {'arc})$
$\to ({'arc}, {'leaf})\ TREEE + ({'arc}, {'leaf})\ TREEE$

$\rightarrow ('arc, 'leaf)\ TREEE$

MkArcTreee $'arc \rightarrow ('arc, 'leaf)\ TREEE \rightarrow ('arc, 'leaf)\ TREEE$

MkTagTreee $(\mathbb{N} \rightarrow 'arc)$
$\rightarrow \mathbb{N}$
$\rightarrow ('arc, 'leaf)\ TREEE$
$\rightarrow ('arc, 'leaf)\ TREEE$

IsTagTreee $(\mathbb{N} \rightarrow 'arc) \rightarrow \mathbb{N} \rightarrow ('arc, 'leaf)\ TREEE \rightarrow BOOL$

UnTagTreee $('arc, 'leaf)\ TREEE \rightarrow ('arc, 'leaf)\ TREEE$

MkListTreee $(\mathbb{N} \rightarrow 'arc)$
$\rightarrow ('arc, 'leaf)\ TREEE\ LIST$
$\rightarrow ('arc, 'leaf)\ TREEE$

AiNToN $\mathbb{N} \rightarrow \mathbb{N}$

AiBoolToN $BOOL \rightarrow \mathbb{N}$

AiOneToN $ONE \rightarrow \mathbb{N}$

NTreeeTag $\mathbb{N} \rightarrow ('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE) \rightarrow 'a \rightarrow (\mathbb{N}, 'leaf)\ TREEE$

NTreeeIsTag $\mathbb{N} \rightarrow (\mathbb{N}, 'leaf)\ TREEE \rightarrow BOOL$

NTreeeMkList $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE) \rightarrow 'a\ LIST \rightarrow (\mathbb{N}, 'leaf)\ TREEE$

NTreeeMkSum $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE)$
$\rightarrow ('b \rightarrow (\mathbb{N}, 'leaf)\ TREEE)$
$\rightarrow 'a + 'b$
$\rightarrow (\mathbb{N}, 'leaf)\ TREEE$

NTreeeMkProd $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE)$
$\rightarrow ('b \rightarrow (\mathbb{N}, 'leaf)\ TREEE)$
$\rightarrow 'a \times 'b$
$\rightarrow (\mathbb{N}, 'leaf)\ TREEE$

NTrListC $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P})$
$\rightarrow 'a\ LIST$
$\rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P}$

NTrSumC $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P})$
$\rightarrow ('b \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P})$
$\rightarrow 'a + 'b$
$\rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P}$

NTrProdC $('a \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P})$
$\rightarrow ('b \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P})$
$\rightarrow 'a \times 'b$
$\rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P}$

NTrLeafC $'a \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P}$

NTreeeTagC $\mathbb{N} \rightarrow (\mathbb{N}, 'leaf)\ TREEE \rightarrow (\mathbb{N}, 'leaf)\ TREEE\ \mathbb{P}$

CR $('b \rightarrow 'a) \rightarrow ('b \rightarrow 'a\ \mathbb{P}) \rightarrow ('b \rightarrow BOOL) \rightarrow 'a \rightarrow 'a \rightarrow BOOL$

FR $((\mathbb{N}, 'a)\ TREEE \rightarrow (\mathbb{N}, 'a)\ TREEE \rightarrow BOOL)\ LIST\ LIST$
$\rightarrow (\mathbb{N}, 'a)\ TREEE\ \mathbb{P}$

14.4 Types

$('1, '2)$ ***CCP***
$('1, '2)$ ***TREEE***

14.5 Fixity

Right Infix 300:

ClosedUnder ***OpenUnder*** $\leq_v$

ClosedUnderFun ***OpenUnderFun***
ClosedUnderRel ***OpenUnderRel***

14.6 Definitions

IsLb $\vdash \forall\ r\ s\ e \bullet\ IsLb\ r\ s\ e \Leftrightarrow (\forall\ x \bullet\ x \in s \Rightarrow r\ e\ x)$

IsUb $\vdash \forall\ r\ s\ e \bullet\ IsUb\ r\ s\ e \Leftrightarrow (\forall\ x \bullet\ x \in s \Rightarrow r\ x\ e)$

IsGlb $\vdash \forall\ r\ s\ e$
$\bullet\ IsGlb\ r\ s\ e$
$\Leftrightarrow IsLb\ r\ s\ e \wedge (\forall\ x \bullet\ IsLb\ r\ s\ x \Rightarrow r\ x\ e)$

IsLub $\vdash \forall\ r\ s\ e$
$\bullet\ IsLub\ r\ s\ e$
$\Leftrightarrow IsUb\ r\ s\ e \wedge (\forall\ x \bullet\ IsUb\ r\ s\ x \Rightarrow r\ e\ x)$

Lub $\vdash \forall\ r\ s \bullet\ Lub\ r\ s = (\epsilon\ x \bullet\ IsLub\ r\ s\ x)$

Glb $\vdash \forall\ r\ s \bullet\ Glb\ r\ s = (\epsilon\ x \bullet\ IsGlb\ r\ s\ x)$

NeGlbsExist $\vdash \forall\ r$
$\bullet\ NeGlbsExist\ r$
$\Leftrightarrow (\forall\ s \bullet\ (\exists\ d \bullet\ d \in s) \Rightarrow (\exists\ e \bullet\ IsGlb\ r\ s\ e))$

GlbsExist $\vdash \forall\ r \bullet\ GlbsExist\ r \Leftrightarrow (\forall\ s \bullet\ \exists\ e \bullet\ IsGlb\ r\ s\ e)$

LubsExist $\vdash \forall\ r \bullet\ LubsExist\ r \Leftrightarrow (\forall\ s \bullet\ \exists\ e \bullet\ IsLub\ r\ s\ e)$

Monotonic $\vdash \forall\ f \bullet\ Monotonic\ f \Leftrightarrow (\forall\ x\ y \bullet\ x \subseteq y \Rightarrow f\ x \subseteq f\ y)$

ClosedUnder $\vdash \forall\ f\ X \bullet\ X\ ClosedUnder\ f \Leftrightarrow f\ X \subseteq X$

OpenUnder $\vdash \forall\ f\ X \bullet\ X\ OpenUnder\ f \Leftrightarrow X \subseteq f\ X$

IsLfp $\vdash \forall\ r\ f\ e$
$\bullet\ IsLfp\ r\ f\ e$
$\Leftrightarrow (let\ fp = \{x | f\ x = x\}\ in\ e \in fp \wedge IsGlb\ r\ fp\ e)$

IsGfp $\vdash \forall\ r\ f\ e$
$\bullet\ IsGfp\ r\ f\ e$
$\Leftrightarrow (let\ fp = \{x | f\ x = x\}\ in\ e \in fp \wedge IsLub\ r\ fp\ e)$

Lfp $\vdash \forall\ f \bullet\ Lfp\ f = \bigcap \{X | X\ ClosedUnder\ f\}$

Gfp $\vdash \forall\ f \bullet\ Gfp\ f = \bigcup \{X | X\ OpenUnder\ f\}$

Rpo $\vdash \forall\ r \bullet\ Rpo\ r \Leftrightarrow PartialOrder\ r \wedge Refl\ r$

RpoU $\vdash \forall\ r \bullet\ RpoU\ r \Leftrightarrow Rpo\ (Universe, r)$

Lfp$_c$ $\vdash \forall\ r\ f \bullet\ (\exists\ e \bullet\ IsLfp\ r\ f\ e) \Rightarrow IsLfp\ r\ f\ (Lfp_c\ r\ f)$

Gfp$_c$ $\vdash \forall\ r\ f \bullet\ (\exists\ e \bullet\ IsGfp\ r\ f\ e) \Rightarrow IsGfp\ r\ f\ (Gfp_c\ r\ f)$

Directed $\vdash \forall\ r\ s$
$\bullet\ Directed\ r\ s$
$\Leftrightarrow (\forall\ x\ y$
$\bullet\ x \in s \wedge y \in s \Rightarrow (\exists\ z \bullet\ z \in s \wedge IsUb\ r\ \{x; y\}\ z))$

DirectedUb $\vdash \forall\ r$
$\bullet\ DirectedUb\ r$
$\Leftrightarrow (\forall\ s \bullet\ Directed\ r\ s \Rightarrow (\exists\ x \bullet\ IsUb\ r\ s\ x))$

Increasing $\vdash \forall\ r\ s\ f$
$\bullet\ Increasing\ r\ s\ f \Leftrightarrow (\forall\ x\ y \bullet\ r\ x\ y \Rightarrow s\ (f\ x)\ (f\ y))$

ChainComplete $\vdash \forall\ X\ r$
$\bullet\ ChainComplete\ (X, r)$
$\Leftrightarrow (\forall\ Y$
$\bullet\ Y \subseteq X \wedge LinearOrder\ (Y, r)$
$\Rightarrow (\exists\ x \bullet\ x \in X \wedge IsLub\ r\ Y\ x))$

CcRpo $\vdash \forall\ r \bullet\ CcRpo\ r \Leftrightarrow Rpo\ r \wedge ChainComplete\ r$

CcRpoU $\vdash \forall\ r\bullet\ CcRpoU\ r \Leftrightarrow CcRpo\ (Universe,\ r)$

CRpo $\vdash \forall\ r\bullet\ CRpo\ r \Leftrightarrow Rpo\ (Universe,\ r) \wedge GlbsExist\ r$

FClosed $\vdash \forall\ f\ X\bullet\ FClosed\ f\ X \Leftrightarrow (\forall\ x\bullet\ x \in X \Rightarrow f\ x \in X)$

FChainClosed $\vdash \forall\ f\ X\ r$
$\bullet\ FChainClosed\ f\ (X,\ r)$
$\Leftrightarrow FClosed\ f\ X \wedge ChainComplete\ (X,\ r)$

FChain $\vdash \forall\ f\ X\ r$
$\bullet\ FChain\ f\ (X,\ r)$
$= \bigcap \{Y | Y \subseteq X \wedge FChainClosed\ f\ (Y,\ r)\}$

FChainU $\vdash \forall\ f\ r\bullet\ FChainU\ f\ r = FChain\ f\ (Universe,\ r)$

Extreme $\vdash \forall\ f\ X\ \$\leq_v$
$\bullet\ Extreme\ (f,\ X,\ \$\leq_v)$
$= \{x$
$|x \in FChain\ f\ (X,\ \$\leq_v)$
$\wedge (\forall\ y$
$\bullet\ y \in FChain\ f\ (X,\ \$\leq_v)$
$\Rightarrow y \leq_v x \wedge \neg\ y = x$
$\Rightarrow f\ y \leq_v x)\}$

S $\vdash \forall\ f\ X\ \$\leq_v\ x$
$\bullet\ S\ (f,\ X,\ \$\leq_v)\ x$
$= \{y$
$|y \in FChain\ f\ (X,\ \$\leq_v) \wedge (y \leq_v x \vee f\ x \leq_v y)\}$

RelInv $\vdash \forall\ r\bullet\ RelInv\ r = (\lambda\ x\ y\bullet\ r\ y\ x)$

ChainCoComplete $\vdash \forall\ X\ r$
$\bullet\ ChainCoComplete\ (X,\ r)$
$\Leftrightarrow (\forall\ Y$
$\bullet\ Y \subseteq X \wedge LinearOrder\ (Y,\ r)$
$\Rightarrow (\exists\ x\bullet\ x \in X \wedge IsGlb\ r\ Y\ x))$

CoCcRpo $\vdash \forall\ r\bullet\ CoCcRpo\ r \Leftrightarrow Rpo\ r \wedge ChainCoComplete\ r$

CoCcRpoU $\vdash \forall\ r\bullet\ CoCcRpoU\ r \Leftrightarrow CoCcRpo\ (Universe,\ r)$

FCoChainClosed $\vdash \forall\ f\ X\ r$
$\bullet\ FCoChainClosed\ f\ (X,\ r)$
$\Leftrightarrow FClosed\ f\ X \wedge ChainCoComplete\ (X,\ r)$

FCoChain $\vdash \forall\ f\ X\ r$
$\bullet\ FCoChain\ f\ (X,\ r)$
$= \bigcap \{Y | Y \subseteq X \wedge FCoChainClosed\ f\ (Y,\ r)\}$

Closed$_c$ $\vdash \forall\ \$\leq_v\ f\ x\bullet\ Closed_c\ (\$\leq_v,\ f)\ x \Leftrightarrow f\ x \leq_v x$

Closure$_c$ $\vdash \forall\ \$\leq_v\ f\ x$
$\bullet\ Closure_c\ (\$\leq_v,\ f)\ x$
$= Glb\ \$\leq_v\ \{y | x \leq_v y \wedge Closed_c\ (\$\leq_v,\ f)\ y\}$

Closure$_d$ $\vdash \forall\ \$\leq_v\ f\ x$
$\bullet\ Closure_d\ (\$\leq_v,\ f)\ x$
$= Lub\ \$\leq_v\ (FChain\ f\ (\{y | x \leq_v y\},\ \$\leq_v))$

CRpoU $\vdash \forall\ r\bullet\ CRpoU\ r \Leftrightarrow RpoU\ r \wedge LubsExist\ r \wedge GlbsExist\ r$

ContProp $\vdash \forall\ r\ p$
$\bullet\ ContProp\ r\ p$
$\Leftrightarrow (\forall\ s\bullet\ (\forall\ y\bullet\ y \in s \Rightarrow p\ y) \Rightarrow p\ (Lub\ r\ s))$

Rising $\vdash \forall\ f\ r\bullet\ Rising\ f\ r = (\lambda\ x\bullet\ r\ x\ (f\ x))$

Falling $\vdash \forall\ f\ r\bullet\ Falling\ f\ r = (\lambda\ x\bullet\ r\ (f\ x)\ x)$

ContProp3 $\vdash \forall\ f\ r\ p$
$\bullet\ ContProp3\ f\ r\ p$
$\Leftrightarrow (\forall\ x$
$\bullet\ (\forall\ y\bullet\ Rising\ f\ r\ y \wedge r\ y\ x \wedge \neg\ r\ x\ y \Rightarrow p\ y)$
$\Rightarrow p\ x)$

Fun2MonoFun $\vdash \forall\ f\ s\bullet\ Fun2MonoFun\ f\ s = \{x | \exists\ t\bullet\ t \subseteq s \wedge x \in f\ t\}$

HeredFun $\vdash \forall\ f\bullet\ HeredFun\ f = Lfp\ (Fun2MonoFun\ f)$

CoHeredFun $\vdash \forall\ f\bullet\ CoHeredFun\ f = Gfp\ (Fun2MonoFun\ f)$

ClosedUnderFun
$\vdash \forall\ s\ f$
$\bullet\ s\ ClosedUnderFun\ f \Leftrightarrow s\ ClosedUnder\ Fun2MonoFun\ f$

OpenUnderFun $\vdash \forall\ s\ f\bullet\ s\ OpenUnderFun\ f \Leftrightarrow s\ OpenUnder\ Fun2MonoFun\ f$

Rules2Fun $\vdash \forall\ r\bullet\ Rules2Fun\ r = (\lambda\ s\bullet\ \{x | (s,\ x) \in r\})$

Rel2Fun $\vdash \forall\ r\bullet\ Rel2Fun\ r = (\lambda\ s\bullet\ \{x | \{y | r\ y\ x\} = s\})$

HeredRel $\vdash \forall\ r\bullet\ HeredRel\ r = HeredFun\ (Rel2Fun\ r)$

CoHeredRel $\vdash \forall\ r\bullet\ CoHeredRel\ r = CoHeredFun\ (Rel2Fun\ r)$

ClosedUnderRel
$\vdash \forall\ s\ r$
$\bullet\ s\ ClosedUnderRel\ r \Leftrightarrow s\ ClosedUnderFun\ Rel2Fun\ r$

OpenUnderRel $\vdash \forall\ s\ r\bullet\ s\ OpenUnderRel\ r \Leftrightarrow s\ OpenUnderFun\ Rel2Fun\ r$

Prop2Rel $\vdash \forall\ (X,\ \$<<)\ p$
$\bullet\ Prop2Rel\ (X,\ \$<<)\ p = (\lambda\ x\ y\bullet\ x << y \wedge p\ y)$

Hereditary $\vdash \forall\ (X,\ \$<<)\ p$
$\bullet\ Hereditary\ (X,\ \$<<)\ p$
$= HeredRel\ (Prop2Rel\ (X,\ \$<<)\ p)$

CoHereditary $\vdash \forall\ (X,\ \$<<)\ p$
$\bullet\ CoHereditary\ (X,\ \$<<)\ p$
$= CoHeredRel\ (Prop2Rel\ (X,\ \$<<)\ p)$

ClosedUnderProp
$\vdash \forall\ (X,\ \$<<)\ s\ p$
$\bullet\ ClosedUnderProp\ (X,\ \$<<)\ s\ p$
$\Leftrightarrow s\ ClosedUnderRel\ Prop2Rel\ (X,\ \$<<)\ p$

OpenUnderProp
$\vdash \forall\ (X,\ \$<<)\ s\ p$
$\bullet\ OpenUnderProp\ (X,\ \$<<)\ s\ p$
$\Leftrightarrow s\ OpenUnderRel\ Prop2Rel\ (X,\ \$<<)\ p$

CCP $\vdash \exists\ f\bullet\ TypeDefn\ (\lambda\ x\bullet\ T)\ f$

MkCCP

Constructor

Content

Predicate $\vdash \forall\ t\ x1\ x2\ x3$
$\bullet\ Constructor\ (MkCCP\ x1\ x2\ x3) = x1$
$\wedge\ Content\ (MkCCP\ x1\ x2\ x3) = x2$
$\wedge\ Predicate\ (MkCCP\ x1\ x2\ x3) = x3$
$\wedge\ MkCCP\ (Constructor\ t)\ (Content\ t)\ (Predicate\ t)$
$= t$

CCP2Fun $\vdash \forall\ ccp\ x$
$\bullet\ CCP2Fun\ ccp\ x$
$= \{y$
$| \exists\ b$
$\bullet\ y = Constructor\ ccp\ b$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad \wedge\ x = Content\ ccp\ b \\
\qquad\qquad\qquad\qquad \wedge\ Predicate\ ccp\ b\} \\
\textbf{\textit{CCP2Rel}} \quad \vdash \forall\ ccp\ x\ y \\
\qquad \bullet\ CCP2Rel\ ccp\ x\ y \\
\qquad\qquad \Leftrightarrow (\exists\ b \\
\qquad\qquad \bullet\ y = Constructor\ ccp\ b \\
\qquad\qquad\qquad \wedge\ x \in Content\ ccp\ b \\
\qquad\qquad\qquad \wedge\ Predicate\ ccp\ b) \\
\textbf{\textit{MapNFun}} \quad \vdash \forall\ tf\ al \\
\qquad \bullet\ MapNFun\ tf\ al \\
\qquad\qquad = Map\ (Uncurry\ tf)\ (Combine\ (1\ ..{}_L\ \#\ al)\ al) \\
\textbf{\textit{MapTag}} \quad \vdash \forall\ tf\ al \\
\qquad \bullet\ MapTag\ tf\ al = MapNFun\ (\lambda\ n\ f\ a\bullet\ tf\ n\ (f\ a))\ al \\
\textbf{\textit{LiftTag}} \quad \vdash \forall\ tf\ n\ s \\
\qquad \bullet\ LiftTag\ tf\ n\ s = \{x | \exists\ y\bullet\ y \in s \wedge x = tf\ n\ y\} \\
\textbf{\textit{CompoundFuns}} \qquad\qquad \vdash \forall\ tf\ asfll \\
\qquad \bullet\ CompoundFuns\ tf\ asfll \\
\qquad\qquad = ListFunUnion \\
\qquad\qquad\ (Map \\
\qquad\qquad\qquad ListFunUnion \\
\qquad\qquad\qquad (Map\ (MapTag\ (LiftTag\ tf))\ asfll)) \\
\textbf{\textit{HCF}} \quad \vdash \forall\ tf\ asfll \\
\qquad \bullet\ HCF\ tf\ asfll = HeredFun\ (CompoundFuns\ tf\ asfll) \\
\textbf{\textit{LiftTag2}} \quad \vdash \forall\ tf\ n\ r \\
\qquad \bullet\ LiftTag2\ tf\ n\ r = (\lambda\ x\ y\bullet \exists\ z\bullet\ r\ x\ z \wedge y = tf\ n\ z) \\
\textbf{\textit{MapTag2}} \quad \vdash \forall\ tf\ rl \\
\qquad \bullet\ MapTag2\ tf\ rl \\
\qquad\qquad = MapNFun\ (\lambda\ n\ r\ x\ y\bullet \exists\ z\bullet\ y = tf\ n\ z \wedge r\ x\ z)\ rl \\
\textbf{\textit{ListRelUnion}} \quad \vdash \forall\ l\ x\ y\bullet\ ListRelUnion\ l\ x\ y \Leftrightarrow (\exists\ r\bullet\ r \in_L l \wedge r\ x\ y) \\
\textbf{\textit{CompoundRels}} \qquad\qquad \vdash \forall\ tf\ arll \\
\qquad \bullet\ CompoundRels\ tf\ arll \\
\qquad\qquad = ListRelUnion \\
\qquad\qquad\ (Map\ ListRelUnion\ (Map\ (MapTag2\ tf)\ arll)) \\
\textbf{\textit{HCR}} \quad \vdash \forall\ tf\ asrll \\
\qquad \bullet\ HCR\ tf\ asrll = HeredRel\ (CompoundRels\ tf\ asrll) \\
\textbf{\textit{HereditarilyPQ}} \\
\qquad\qquad \vdash \forall\ c\ d\ P\ Q \\
\qquad\qquad \bullet\ HereditarilyPQ\ (c,\ d,\ P,\ Q) \\
\qquad\qquad\qquad = (\bigcap \\
\qquad\qquad\qquad\qquad \{X' \\
\qquad\qquad\qquad\qquad\ |\forall\ x \\
\qquad\qquad\qquad\qquad\qquad \bullet\ P\ x \\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ (\forall\ y\ z \\
\qquad\qquad\qquad\qquad\qquad\qquad \bullet\ y \in c\ x \wedge z \in d\ y \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Rightarrow Q\ y \wedge z \in X') \\
\qquad\qquad\qquad\qquad\qquad\qquad \Rightarrow x \in X'\}, \\
\qquad\qquad\qquad\ \bigcap \\
\qquad\qquad\qquad\qquad \{Y' \\
\qquad\qquad\qquad\qquad\ |\forall\ x \\
\qquad\qquad\qquad\qquad\qquad \bullet\ Q\ x \\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ (\forall\ y\ z
\end{array}
$$

$$\bullet\ y \in d\ x \wedge z \in c\ y \Rightarrow P\ y \wedge z \in Y')\ \Rightarrow x \in Y'\})$$

I_closed $\vdash \forall\ r\ w \bullet\ I\_closed\ r\ w \Leftrightarrow (\forall\ x \bullet (\exists\ y \bullet\ r\ y\ x \wedge (\forall\ z \bullet\ r\ z\ y \Rightarrow z \in w)) \Rightarrow x \in w)$

II_closed $\vdash \forall\ r\ w \bullet\ II\_closed\ r\ w \Leftrightarrow (\forall\ x \bullet (\forall\ y \bullet\ r\ y\ x \Rightarrow (\exists\ z \bullet\ r\ z\ y \wedge z \in w)) \Rightarrow x \in w)$

I_closure $\vdash \forall\ r \bullet\ I\_closure\ r = \bigcap \{x | I\_closed\ r\ x\}$

II_closure $\vdash \forall\ r \bullet\ II\_closure\ r = \bigcap \{x | II\_closed\ r\ x\}$

PseudoWellFounded $\vdash \forall\ r \bullet\ PseudoWellFounded\ r \Leftrightarrow I\_closure\ r \cup II\_closure\ r = Universe$

P_closed $\vdash \forall\ c\ d\ P\ Q\ w \bullet\ P\_closed\ (c,\ d,\ P,\ Q)\ w \Leftrightarrow (\forall\ x \bullet\ P\ x \wedge (\forall\ y \bullet\ y \in c\ x \Rightarrow Q\ y \wedge (\exists\ z \bullet\ z \in d\ y \wedge z \in w))\ \Rightarrow x \in w)$

Q_closed $\vdash \forall\ c\ d\ P\ Q\ w \bullet\ Q\_closed\ (c,\ d,\ P,\ Q)\ w \Leftrightarrow (\forall\ x \bullet\ Q\ x \wedge (\exists\ y \bullet\ y \in d\ x \wedge P\ y \wedge (\forall\ z \bullet\ z \in c\ y \Rightarrow z \in w))\ \Rightarrow x \in w)$

P_closure $\vdash \forall\ z \bullet\ P\_closure\ z = \bigcap \{x | P\_closed\ z\ x\}$

Q_closure $\vdash \forall\ z \bullet\ Q\_closure\ z = \bigcap \{x | Q\_closed\ z\ x\}$

PseudoWellFoundedPQ $\vdash \forall\ z \bullet\ PseudoWellFoundedPQ\ z \Leftrightarrow P\_closure\ z \cap Q\_closure\ z = \{\} \wedge P\_closure\ z \cup Q\_closure\ z = Universe$

PseudoHereditarilyPQ $\vdash \forall\ z \bullet\ PseudoHereditarilyPQ\ z = (P\_closure\ z,\ Q\_closure\ z)$

FunEquiv $\vdash \forall\ f\ g\ X \bullet\ FunEquiv\ f\ g\ X \Leftrightarrow (\forall\ x \bullet\ x \in X \Rightarrow g\ x = f\ x)$

FixedRegion $\vdash \forall\ G\ f\ X \bullet\ FixedRegion\ G\ f\ X \Leftrightarrow (\forall\ g \bullet (\forall\ x \bullet\ x \in X \Rightarrow g\ x = f\ x) \Rightarrow (\forall\ x \bullet\ x \in X \Rightarrow G\ g\ x = f\ x))$

LiftProduct $\vdash \forall\ prod\ f\ g\ a\ b \bullet\ LiftProduct\ prod\ f\ g\ (a,\ b) = prod\ (f\ a,\ g\ b)$

LiftSum $\vdash \forall\ sum\ f\ g\ ab \bullet LiftSum\ sum\ f\ g\ ab = sum\ (Fun_+\ f\ g\ ab)$

LiftList $\vdash \forall\ list\ f\ al \bullet LiftList\ list\ f\ al = list\ (Map\ f\ al)$

LiftSumUnion $\vdash \forall\ cfl\ cfr\ ab$
$\bullet\ LiftSumUnion\ cfl\ cfr\ ab$
$= (if\ IsL\ ab$
$then\ cfl\ (OutL\ ab)$
$else\ cfr\ (OutR\ ab))$

LiftSumPred $\vdash \forall\ pl\ pr\ ab$
$\bullet\ LiftSumPred\ pl\ pr\ ab$
$\Leftrightarrow (if\ IsL\ ab\ then\ pl\ (OutL\ ab)\ else\ pr\ (OutR\ ab))$

TREEE $\vdash \exists\ f \bullet TypeDefn\ (\lambda\ x \bullet T)\ f$

MkTREEE

Treee $\vdash \forall\ t\ x1$
$\bullet\ Treee\ (MkTREEE\ x1) = x1 \wedge MkTREEE\ (Treee\ t) = t$

Domain $\vdash \forall\ r \bullet Domain\ r = \{x | \exists\ y \bullet r\ x\ y\}$

NullTreee $\vdash NullTreee = MkTREEE\ (\lambda\ al\ n \bullet F)$

IsTreee $\vdash \forall\ t$
$\bullet\ IsTreee\ t$
$\Leftrightarrow \neg\ t = NullTreee$
$\wedge\ ManyOne\ (Treee\ t)$
$\wedge (\forall\ p\ q$
$\bullet\ p \in Domain\ (Treee\ t)$
$\Rightarrow \neg\ p\ @\ q \in Domain\ (Treee\ t))$

MkTreee $\vdash \forall\ c$
$\bullet\ MkTreee\ c$
$= MkTREEE$
$(\lambda\ path\ leaf$
$\bullet \exists\ tr$
$\bullet\ c\ (Head\ path)\ tr$
$\wedge\ Treee\ tr\ (Tail\ path)\ leaf)$

NiceChildren $\vdash \forall\ c$
$\bullet\ NiceChildren\ c \Leftrightarrow ManyOne\ c \wedge \neg\ (\exists\ a \bullet\ c\ a\ NullTreee)$

DestTreee $\vdash \forall\ c \bullet NiceChildren\ c \Rightarrow DestTreee\ (MkTreee\ c) = c$

MkLeafTreee $\vdash \forall\ l$
$\bullet\ MkLeafTreee\ l$
$= MkTREEE\ (\lambda\ arc\ leaf \bullet arc = [] \wedge leaf = l)$

MkProdTreee $\vdash \forall\ ai\ l\ r$
$\bullet\ MkProdTreee\ ai\ (l, r)$
$= MkTreee$
$(\lambda\ arc\ tree$
$\bullet\ arc = ai\ T \wedge tree = l$
$\vee\ arc = ai\ F \wedge tree = r)$

MkSumTreee $\vdash \forall\ ai\ t$
$\bullet\ MkSumTreee\ ai\ t$
$= MkTreee$
$(\lambda\ arc\ tree$
$\bullet\ if\ IsL\ t$
$then\ arc = ai\ T \wedge tree = OutL\ t$
$else\ arc = ai\ F \wedge tree = OutR\ t)$

MkArcTreee $\vdash \forall\ a\ t$
$\bullet\ MkArcTreee\ a\ t$

$= MkTreee\ (\lambda\ arc\ tree \bullet\ arc = a \wedge tree = t)$

MkTagTreee $\vdash \forall\ ai\ n\ t \bullet\ MkTagTreee\ ai\ n\ t = MkArcTreee\ (ai\ n)\ t$

IsTagTreee $\vdash \forall\ ai\ n\ t$
$\bullet\ IsTagTreee\ ai\ n\ t \Leftrightarrow (\exists\ t2 \bullet\ t = MkTagTreee\ ai\ n\ t2)$

UnTagTreee $\vdash ConstSpec$
$(\lambda\ UnTagTreee'$
$\bullet \forall\ t$
$\bullet \exists\ t2\ a$
$\bullet\ t = MkArcTreee\ a\ t2 \Rightarrow UnTagTreee'\ t = t2)$
$UnTagTreee$

MkListTreee $\vdash \forall\ ai\ trl$
$\bullet\ MkListTreee\ ai\ trl$
$= MkTreee$
$(\lambda\ arc\ tree$
$\bullet \exists\ n \bullet\ arc = ai\ n \wedge (n,\ tree) \in ListRel\ trl)$

AiOneToN

AiBoolToN

AiNToN $\vdash (\forall\ one \bullet\ AiOneToN\ one = 0)$
$\wedge (\forall\ b \bullet\ AiBoolToN\ b = (if\ b\ then\ 1\ else\ 0))$
$\wedge (\forall\ n \bullet\ AiNToN\ n = n)$

NTreeeTag $\vdash \forall\ n\ f \bullet\ NTreeeTag\ n\ f = MkTagTreee\ AiNToN\ n\ o\ f$

NTreeeIsTag $\vdash NTreeeIsTag = IsTagTreee\ AiNToN$

NTreeeMkList $\vdash NTreeeMkList = LiftList\ (MkListTreee\ AiNToN)$

NTreeeMkProd

NTreeeMkSum $\vdash NTreeeMkProd = LiftProduct\ (MkProdTreee\ AiBoolToN)$
$\wedge\ NTreeeMkSum = LiftSum\ (MkSumTreee\ AiBoolToN)$

NTrListC $\vdash NTrListC = LiftList\ (\lambda\ sl \bullet \bigcup \{x | x \in_L sl\})$

NTrProdC

NTrSumC $\vdash NTrProdC = LiftProduct\ (Uncurry\ \$\cup)$
$\wedge\ NTrSumC$
$= LiftSum\ (\lambda\ x \bullet\ if\ IsL\ x\ then\ OutL\ x\ else\ OutR\ x)$

NTrLeafC $\vdash \forall\ l \bullet\ NTrLeafC\ l = \{\}$

NTreeeTagC $\vdash \forall\ n\ t \bullet\ NTreeeTagC\ n\ t = \{MkTagTreee\ AiNToN\ n\ t\}$

CR $\vdash \forall\ tor\ tent\ pred$
$\bullet\ CR\ tor\ tent\ pred = CCP2Rel\ (MkCCP\ tor\ tent\ pred)$

FR $\vdash \forall\ rll$
$\bullet\ FR\ rll$
$= HeredRel\ (CompoundRels\ (MkTagTreee\ AiNToN)\ rll)$

14.7 Theorems

lub_unique_lemma
$\vdash \forall\ X\ r\ x\ y$
$\bullet\ Antisym\ (Universe,\ r)$
$\Rightarrow IsLub\ r\ X\ x \wedge IsLub\ r\ X\ y$
$\Rightarrow x = y$

lub_unique_lemma2
$\vdash \forall\ X\ r\ x\ y$
$\bullet\ Antisym\ (X,\ r) \wedge x \in X \wedge y \in X$
$\Rightarrow IsLub\ r\ X\ x \wedge IsLub\ r\ X\ y$
$\Rightarrow x = y$

glb_unique_lemma

$$\vdash \forall\ X\ r\ x\ y \bullet\ Antisym\ (Universe,\ r) \Rightarrow IsGlb\ r\ X\ x \wedge IsGlb\ r\ X\ y \Rightarrow x = y$$

glb_unique_lemma2

$$\vdash \forall\ X\ r\ x\ y \bullet\ Antisym\ (X,\ r) \wedge x \in X \wedge y \in X \Rightarrow IsGlb\ r\ X\ x \wedge IsGlb\ r\ X\ y \Rightarrow x = y$$

isub_sub_lemma

$$\vdash \forall\ X\ Y\ r\ y \bullet\ X \subseteq Y \wedge IsUb\ r\ Y\ y \Rightarrow IsUb\ r\ X\ y$$

islb_sub_lemma

$$\vdash \forall\ X\ Y\ r\ y \bullet\ X \subseteq Y \wedge IsLb\ r\ Y\ y \Rightarrow IsLb\ r\ X\ y$$

isub_trans_lemma

$$\vdash \forall\ X\ r\ x\ y \bullet\ Trans\ (Universe,\ r) \wedge IsUb\ r\ X\ x \wedge r\ x\ y \Rightarrow IsUb\ r\ X\ y$$

islb_trans_lemma

$$\vdash \forall\ X\ r\ x\ y \bullet\ Trans\ (Universe,\ r) \wedge IsLb\ r\ X\ y \wedge r\ x\ y \Rightarrow IsLb\ r\ X\ x$$

islub_sub_lemma

$$\vdash \forall\ X\ Y\ r\ x\ y \bullet\ X \subseteq Y \wedge IsLub\ r\ X\ x \wedge IsLub\ r\ Y\ y \Rightarrow r\ x\ y$$

isglb_sub_lemma

$$\vdash \forall\ X\ Y\ r\ x\ y \bullet\ X \subseteq Y \wedge IsGlb\ r\ X\ x \wedge IsGlb\ r\ Y\ y \Rightarrow r\ y\ x$$

lub_lub_lemma

$$\vdash \forall\ X\ r \bullet\ (\exists\ z \bullet\ IsLub\ r\ X\ z) \Rightarrow IsLub\ r\ X\ (Lub\ r\ X)$$

glb_glb_lemma

$$\vdash \forall\ X\ r \bullet\ (\exists\ z \bullet\ IsGlb\ r\ X\ z) \Rightarrow IsGlb\ r\ X\ (Glb\ r\ X)$$

isglb_glb_lemma

$$\vdash \forall\ X\ r\ z \bullet\ Antisym\ (Universe,\ r) \Rightarrow IsGlb\ r\ X\ z \Rightarrow Glb\ r\ X = z$$

islub_lub_lemma

$$\vdash \forall\ X\ r\ z \bullet\ Antisym\ (Universe,\ r) \Rightarrow IsLub\ r\ X\ z \Rightarrow Lub\ r\ X = z$$

ub_ub_lemma1 $\vdash \forall\ r\ X\ y \bullet\ IsUb\ r\ X\ y \Rightarrow (\forall\ x \bullet\ x \in X \Rightarrow r\ x\ y)$

lub_ub_lemma1

$$\vdash \forall\ r\ X\ y \bullet\ IsLub\ r\ X\ y \Rightarrow (\forall\ x \bullet\ x \in X \Rightarrow r\ x\ y)$$

lb_lb_lemma1 $\vdash \forall\ r\ X\ y \bullet\ IsLb\ r\ X\ y \Rightarrow (\forall\ x \bullet\ x \in X \Rightarrow r\ y\ x)$

glb_lb_lemma1

$$\vdash \forall\ r\ X\ y \bullet\ IsGlb\ r\ X\ y \Rightarrow (\forall\ x \bullet\ x \in X \Rightarrow r\ y\ x)$$

lin_refl_lub_lemma

$$\vdash \forall\ r\ X\ s\ l \bullet\ LinearOrder\ (X,\ r) \wedge Refl\ (X,\ r) \wedge s \subseteq X \wedge l \in X \wedge IsLub\ r\ s\ l$$

$\Rightarrow (\forall\ y$
$\bullet\ y \in X \wedge r\ y\ l \wedge \neg\ r\ l\ y \Rightarrow (\exists\ z\bullet\ z \in s \wedge r\ y\ z))$

lub_lub_lemma2
$\vdash \forall\ r\ X\bullet\ LubsExist\ r \Rightarrow IsLub\ r\ X\ (Lub\ r\ X)$

glb_glb_lemma2
$\vdash \forall\ r\ X\bullet\ GlbsExist\ r \Rightarrow IsGlb\ r\ X\ (Glb\ r\ X)$

less_lub_lemma
$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow (\forall\ x\ X\bullet\ x \in X \Rightarrow r\ x\ (Lub\ r\ X))$

gt_glb_lemma $\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow (\forall\ x\ X\bullet\ x \in X \Rightarrow r\ (Glb\ r\ X)\ x)$

lub_sub_lemma
$\vdash \forall\ r$
$\bullet\ LubsExist\ r$
$\Rightarrow (\forall\ X\ Y\bullet\ X \subseteq Y \Rightarrow r\ (Lub\ r\ X)\ (Lub\ r\ Y))$

glb_sub_lemma
$\vdash \forall\ r$
$\bullet\ GlbsExist\ r$
$\Rightarrow (\forall\ X\ Y\bullet\ X \subseteq Y \Rightarrow r\ (Glb\ r\ Y)\ (Glb\ r\ X))$

le_islub_lub_lemma
$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow (\forall\ X\bullet\ IsLub\ r\ X\ (Lub\ r\ X))$

ge_isglb_glb_lemma
$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow (\forall\ X\bullet\ IsGlb\ r\ X\ (Glb\ r\ X))$

lea_islub_lub_lemma
$\vdash \forall\ r$
$\bullet\ LubsExist\ r \wedge Antisym\ (Universe,\ r)$
$\Rightarrow (\forall\ X\ x\bullet\ IsLub\ r\ X\ x \Leftrightarrow Lub\ r\ X = x)$

gea_isglb_glb_lemma
$\vdash \forall\ r$
$\bullet\ GlbsExist\ r \wedge Antisym\ (Universe,\ r)$
$\Rightarrow (\forall\ X\ x\bullet\ IsGlb\ r\ X\ x \Leftrightarrow Glb\ r\ X = x)$

islfp_lemma1 $\vdash \forall\ r\ f\ e$
$\bullet\ IsLfp\ r\ f\ e \Rightarrow e = f\ e \wedge (\forall\ x\bullet\ x = f\ x \Rightarrow r\ e\ x)$

isgfp_lemma1 $\vdash \forall\ r\ f\ e$
$\bullet\ IsGfp\ r\ f\ e \Rightarrow e = f\ e \wedge (\forall\ x\bullet\ x = f\ x \Rightarrow r\ x\ e)$

islfp_lemma2 $\vdash \forall\ r\ f\ e$
$\bullet\ IsLfp\ r\ f\ e \Rightarrow f\ e = e \wedge (\forall\ x\bullet\ f\ x = x \Rightarrow r\ e\ x)$

isgfp_lemma2 $\vdash \forall\ r\ f\ e$
$\bullet\ IsGfp\ r\ f\ e \Rightarrow f\ e = e \wedge (\forall\ x\bullet\ f\ x = x \Rightarrow r\ x\ e)$

lfp_fixedpoint_thm
$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow h\ (Lfp\ h) = Lfp\ h$

Lfp_lfp_thm $\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ g\bullet\ h\ g = g \Rightarrow Lfp\ h \subseteq g)$

gfp_fixedpoint_thm
$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow h\ (Gfp\ h) = Gfp\ h$

gfp_gfp_thm $\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ g\bullet\ h\ g = g \Rightarrow g \subseteq Gfp\ h)$

lfp_closed_thm
$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow Lfp\ h\ ClosedUnder\ h$

lfp_closed_thm1
$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ x\bullet\ x \in h\ (Lfp\ h) \Rightarrow x \in Lfp\ h)$

lfp_open_thm $\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow Lfp\ h\ OpenUnder\ h$

lfp_open_thm1
$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ x\bullet\ x \in Lfp\ h \Rightarrow x \in h\ (Lfp\ h))$

gfp_closed_thm

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow Gfp\ h\ ClosedUnder\ h$

gfp_closed_thm1

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ x\bullet\ x \in h\ (Gfp\ h) \Rightarrow x \in Gfp\ h)$

gfp_open_thm $\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow Gfp\ h\ OpenUnder\ h$

gfp_open_thm1

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ x\bullet\ x \in Gfp\ h \Rightarrow x \in h\ (Gfp\ h))$

lfp_induction_thm

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ s\bullet\ s\ ClosedUnder\ h \Rightarrow Lfp\ h \subseteq s)$

lfp_induction_thm1

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ s\bullet\ h\ s \subseteq s \Rightarrow Lfp\ h \subseteq s)$

gfp_coinduction_thm

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ s\bullet\ s\ OpenUnder\ h \Rightarrow s \subseteq Gfp\ h)$

gfp_coinduction_thm1

$\vdash \forall\ h\bullet\ Monotonic\ h \Rightarrow (\forall\ s\bullet\ s \subseteq h\ s \Rightarrow s \subseteq Gfp\ h)$

lfp_induction_thm2

$\vdash \forall\ h$
- $Monotonic\ h$
  $\Rightarrow (\forall\ s\bullet\ (s \cap Lfp\ h)\ ClosedUnder\ h \Rightarrow Lfp\ h \subseteq s)$

gfp_coinduction_thm2

$\vdash \forall\ h$
- $Monotonic\ h$
  $\Rightarrow (\forall\ s\bullet\ (s \cup Gfp\ h)\ OpenUnder\ h \Rightarrow s \subseteq Gfp\ h)$

lfp_induction_thm3

$\vdash \forall\ h$
- $Monotonic\ h \Rightarrow (\forall\ s\bullet\ h\ (s \cap Lfp\ h) \subseteq s \Rightarrow Lfp\ h \subseteq s)$

gfp_coinduction_thm3

$\vdash \forall\ h$
- $Monotonic\ h \Rightarrow (\forall\ s\bullet\ s \subseteq h\ (s \cup Gfp\ h) \Rightarrow s \subseteq Gfp\ h)$

rpo_fc_clauses

$\vdash \forall\ (X, \$\leq_v)$
- $Rpo\ (X, \$\leq_v)$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in X \wedge y \in X \wedge x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$
  $\wedge\ (\forall\ x\ y\ z$
  $\bullet\ x \in X \wedge y \in X \wedge z \in X \wedge x \leq_v y \wedge y \leq_v z$
  $\Rightarrow x \leq_v z)$
  $\wedge\ (\forall\ x\bullet\ x \in X \Rightarrow x \leq_v x)$

rpou_fc_clauses

$\vdash \forall\ \$\leq_v$
- $Rpo\ (Universe, \$\leq_v)$
  $\Rightarrow (\forall\ x\ y\bullet\ x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$
  $\wedge\ (\forall\ x\ y\ z\bullet\ x \leq_v y \wedge y \leq_v z \Rightarrow x \leq_v z)$
  $\wedge\ (\forall\ x\bullet\ x \leq_v x)$

rpou_fc_clauses2

$\vdash \forall\ \$\leq_v$
- $RpoU\ \$\leq_v$
  $\Rightarrow (\forall\ x\ y\bullet\ x \leq_v y \wedge y \leq_v x \Rightarrow x = y)$
  $\wedge\ (\forall\ x\ y\ z\bullet\ x \leq_v y \wedge y \leq_v z \Rightarrow x \leq_v z)$
  $\wedge\ (\forall\ x\bullet\ x \leq_v x)$

increasing_funcomp_thm

$\vdash \forall\ r1\ r2\ r3\ f\ g$

$\bullet$ *Increasing r1 r2 f* $\wedge$ *Increasing r2 r3 g*
$\Rightarrow$ *Increasing r1 r3* $(g \circ f)$

increasing_funcomp_thm2
$\vdash \forall$ *r1 r2 r3 f g*
$\bullet$ *Increasing r1 r2 f* $\wedge$ *Increasing r2 r3 g*
$\Rightarrow$ *Increasing r1 r3* $(\lambda\, x\bullet\ g\ (f\ x))$

mono_fixp_thm
$\vdash \forall$ *f r*
$\bullet$ *Refl* (*Universe*, *r*)
$\wedge\ (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$
$\wedge$ *trans r*
$\wedge$ *Increasing r r f*
$\wedge$ *NeGlbsExist r*
$\wedge\ (\exists\ x\bullet\ r\ (f\ x)\ x)$
$\Rightarrow\ (\exists\ e\bullet$ *IsGlb r* $\{x|r\ (f\ x)\ x\}\ e \wedge$ *IsLfp r f e*)

mono_fixp_thm2
$\vdash \forall$ *f r*
$\bullet$ *Refl* (*Universe*, *r*)
$\wedge\ (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$
$\wedge$ *trans r*
$\wedge$ *Increasing r r f*
$\wedge$ *GlbsExist r*
$\Rightarrow\ (\exists\ e\bullet$ *IsGlb r* $\{x|r\ (f\ x)\ x\}\ e \wedge$ *IsLfp r f e*)

mono_fixp_thm3
$\vdash \forall$ *f r*
$\bullet$ *Refl* (*Universe*, *r*)
$\wedge\ (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$
$\wedge$ *trans r*
$\wedge$ *Increasing r r f*
$\wedge$ *LubsExist r*
$\Rightarrow\ (\exists\ e\bullet$ *IsLub r* $\{x|r\ x\ (f\ x)\}\ e \wedge$ *IsGfp r f e*)

cc_lub_lemma $\vdash \forall$ *X r*
$\bullet$ *ChainComplete* (X, r)
$\Rightarrow\ (\forall\ Y$
$\bullet\ Y \subseteq X \wedge$ *LinearOrder* (Y, r)
$\Rightarrow$ *IsLub r Y* $(Lub\ r\ Y))$

ccu_lub_lemma
$\vdash \forall$ *r*
$\bullet$ *ChainComplete* (*Universe*, *r*)
$\Rightarrow\ (\forall\ Y\bullet$ *LinearOrder* $(Y, r) \Rightarrow$ *IsLub r Y* $(Lub\ r\ Y))$

ccrpou_fc_clauses
$\vdash \forall$ *r*
$\bullet$ *CcRpoU r*
$\Rightarrow$ *ChainComplete* (*Universe*, *r*)
$\wedge\ (\forall\ x\bullet\ r\ x\ x)$
$\wedge\ (\forall\ x\ y\ z\bullet\ r\ x\ y \wedge r\ y\ z \Rightarrow r\ x\ z)$
$\wedge\ (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$

ccrpou_lub_lemma
$\vdash \forall$ *r*
$\bullet$ *CcRpoU r*
$\Rightarrow\ (\forall\ Y\bullet$ *LinearOrder* $(Y, r) \Rightarrow$ *IsLub r Y* $(Lub\ r\ Y))$

ccrpou_lub_lemma2

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow (\forall\ Y \bullet\ LinearOrder\ (Y,\ r) \Rightarrow (\forall\ x \bullet\ x \in Y \Rightarrow r\ x\ (Lub\ r\ Y)))$$

ccrpou_lub_unique_lemma

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow (\forall\ Y \bullet\ LinearOrder\ (Y,\ r) \Rightarrow (\forall\ x \bullet\ IsLub\ r\ Y\ x \Rightarrow x = Lub\ r\ Y))$$

fclosed_universe_lemma

$$\vdash \forall\ f \bullet\ FClosed\ f\ Universe$$

ccrpou_fchainclosed_lemma

$$\vdash \forall\ r \bullet\ CcRpoU\ r \Rightarrow FChainClosed\ f\ (Universe,\ r)$$

ccrpou_fchainclosed_lemma2

$$\vdash \forall\ r\ X \bullet\ CcRpoU\ r \wedge FChainClosed\ f\ (X,\ r) \Rightarrow (\forall\ s \bullet\ s \subseteq X \wedge LinearOrder\ (s,\ r) \Rightarrow Lub\ r\ s \in X)$$

fchain_lemma1

$$\vdash \forall\ X\ r\ f \bullet\ FChainClosed\ f\ (X,\ r) \Rightarrow FChain\ f\ (X,\ r) \subseteq X$$

fchain_fchainclosed_lemma

$$\vdash \forall\ X\ r\ f \bullet\ Antisym\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r) \Rightarrow FChainClosed\ f\ (FChain\ f\ (X,\ r),\ r)$$

fchain_fchainclosed_lemma2

$$\vdash \forall\ X\ r\ f \bullet\ Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r) \Rightarrow FChainClosed\ f\ (FChain\ f\ (X,\ r),\ r)$$

ccrpou_fcclosed_fc_lemma

$$\vdash \forall\ r\ f \bullet\ CcRpoU\ r \Rightarrow FChainClosed\ f\ (FChainU\ f\ r,\ r)$$

ccrpou_fcclosed_fc_lemma2

$$\vdash \forall\ r\ f\ X \bullet\ CcRpoU\ r \wedge X \subseteq FChainU\ f\ r \wedge LinearOrder\ (X,\ r) \Rightarrow Lub\ r\ X \in FChainU\ f\ r$$

fc_fcclosed_fc_lemma

$$\vdash \forall\ X\ r\ f \bullet\ Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r) \Rightarrow (\forall\ x \bullet\ x \in FChain\ f\ (X,\ r) \Rightarrow f\ x \in FChain\ f\ (X,\ r))$$

fchain_lemma2

$$\vdash \forall\ X\ Y\ r\ f \bullet\ Rpo\ (Universe,\ r) \wedge FChainClosed\ f\ (X,\ r) \wedge Y \subseteq FChain\ f\ (X,\ r) \wedge FChainClosed\ f\ (Y,\ r) \Rightarrow Y = FChain\ f\ (X,\ r)$$

fchain_lemma3

$$\vdash \forall\ X\ Y\ r\ f \bullet\ CcRpoU\ r \wedge Y \subseteq FChainU\ f\ r \wedge FChainClosed\ f\ (Y,\ r)$$

$\Rightarrow\ Y = FChainU\ f\ r$

fchain_lemma4

$\vdash \forall\ r\ f\ Y$
- $CcRpoU\ r$
  $\wedge\ FClosed\ f\ (Y \cap FChainU\ f\ r)$
  $\wedge\ ChainComplete\ (Y \cap FChainU\ f\ r,\ r)$
  $\Rightarrow FChainU\ f\ r \subseteq Y$

fchainu_lemma1

$\vdash \forall\ f\ r\ x\bullet\ x \in FChainU\ f\ r \Rightarrow f\ x \in FChainU\ f\ r$

fchainu_induction_thm1

$\vdash \forall\ r\ f\ p$
- $CcRpoU\ r$
  $\wedge\ (\forall\ x\bullet\ x \in FChainU\ f\ r \wedge p\ x \Rightarrow p\ (f\ x))$
  $\wedge\ (\forall\ s$
  $\bullet\ s \subseteq FChainU\ f\ r \wedge (\forall\ y\bullet\ y \in s \Rightarrow p\ y)$
  $\Rightarrow p\ (Lub\ r\ s))$
  $\Rightarrow x \in FChainU\ f\ r$
  $\Rightarrow p\ x$

ccrpo_fixp_lemma4

$\vdash \forall\ X\ r\ f$
- $Increasing\ r\ r\ f$
  $\wedge\ Rpo\ (Universe,\ r)$
  $\wedge\ FChainClosed\ f\ (X,\ r)$
  $\Rightarrow (\forall\ z$
  $\bullet\ z \in FChain\ f\ (X,\ r) \Rightarrow r\ z\ (f\ z) \vee z = f\ z)$

ccrpou_fchainu_lemma1

$\vdash \forall\ r\ f$
- $CcRpoU\ r \wedge Increasing\ r\ r\ f$
  $\Rightarrow (\forall\ z\bullet\ z \in FChainU\ f\ r \Rightarrow r\ z\ (f\ z))$

ccrpou_fchainu_lemma2

$\vdash \forall\ \$\leq_v\ f$
- $CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in FChainU\ f\ \$\leq_v \wedge y \in FChainU\ f\ \$\leq_v$
  $\Rightarrow y \leq_v x \wedge \neg\ y = x$
  $\Rightarrow f\ y \leq_v x)$

ccrpou_fchainu_lemma2b

$\vdash \forall\ \$\leq_v\ f$
- $CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in FChainU\ f\ \$\leq_v \wedge y \in FChainU\ f\ \$\leq_v$
  $\Rightarrow y \leq_v x \wedge \neg\ x \leq_v y$
  $\Rightarrow f\ y \leq_v x)$

ccrpou_fchainu_linear_lemma

$\vdash \forall\ \$\leq_v\ f$
- $CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
  $\Rightarrow LinearOrder\ (FChainU\ f\ \$\leq_v,\ \$\leq_v)$

ccrpou_fchainu_linear_lemma2

$\vdash \forall\ X\ \$\leq_v\ f$
- $CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
  $\Rightarrow (\forall\ X$

$\bullet\ X \subseteq FChainU\ f\ \$\leq_v \Rightarrow LinearOrder\ (X, \$\leq_v))$

ccrpou_fchainu_lemma2c

$\vdash \forall\ \$\leq_v\ f$
$\bullet\ CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
$\Rightarrow (\forall\ x\ y$
$\bullet\ x \in FChainU\ f\ \$\leq_v \wedge y \in FChainU\ f\ \$\leq_v$
$\Rightarrow y \leq_v x \vee f\ x \leq_v y)$

fchainu_induction_thm2

$\vdash \forall\ r\ f\ p$
$\bullet\ CcRpoU\ r$
$\wedge\ Increasing\ r\ r\ f$
$\wedge (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\wedge (\forall\ y$
$\bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y$
$\Rightarrow p\ y)$
$\Rightarrow p\ x)$
$\Rightarrow (\forall\ x\bullet\ x \in FChainU\ f\ r \Rightarrow p\ x)$

ccrpou_fchainu_lfp_lemma

$\vdash \forall\ X\ \$\leq_v\ f$
$\bullet\ CcRpoU\ \$\leq_v \wedge Increasing\ \$\leq_v\ \$\leq_v\ f$
$\Rightarrow (\exists\ l$
$\bullet\ IsLub\ \$\leq_v\ (FChainU\ f\ \$\leq_v)\ l \wedge IsLfp\ \$\leq_v\ f\ l)$

ccrpou_fixp_induction_thm

$\vdash \forall\ r\ f\ p$
$\bullet\ CcRpoU\ r$
$\wedge\ Increasing\ r\ r\ f$
$\wedge (\forall\ x\bullet\ p\ x \Rightarrow p\ (f\ x))$
$\wedge (\forall\ s$
$\bullet (\forall\ x\bullet\ x \in s \Rightarrow p\ x) \wedge LinearOrder\ (s, r)$
$\Rightarrow (\exists\ y\bullet\ p\ y \wedge IsLub\ r\ s\ y))$
$\Rightarrow p\ (Lfp_c\ r\ f)$

fchainu_cases_lemma

$\vdash \forall\ r\ f$
$\bullet\ CcRpoU\ r \wedge Increasing\ r\ r\ f$
$\Rightarrow (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\Rightarrow (\exists\ y\bullet\ y \in FChainU\ f\ r \wedge x = f\ y)$
$\vee\ x$
$= Lub$
r
$\{z | z \in FChainU\ f\ r \wedge r\ z\ x \wedge \neg\ r\ x\ z\})$

fchainu_cases_lemma2

$\vdash \forall\ r\ f$
$\bullet\ CcRpoU\ r \wedge Increasing\ r\ r\ f$
$\Rightarrow (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\Rightarrow (\exists\ y\bullet\ y \in FChainU\ f\ r \wedge \neg\ y = x \wedge x = f\ y)$
$\vee\ x$
$= Lub$
r

$\{z|z \in FChainU\ f\ r \wedge r\ z\ x \wedge \neg\ r\ x\ z\})$

fchainu_induction_thm3

$\vdash \forall\ r\ f\ p$
$\bullet\ CcRpoU\ r$
$\wedge\ Increasing\ r\ r\ f$
$\wedge\ (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\wedge\ (\forall\ y$
$\bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ (f\ x)\ y$
$\Rightarrow p\ y)$
$\Rightarrow p\ (f\ x))$
$\wedge\ (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\wedge\ x$
$= Lub$
r
$\{y|y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y\}$
$\wedge\ (\forall\ y$
$\bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y$
$\Rightarrow p\ y)$
$\Rightarrow p\ x)$
$\Rightarrow (\forall\ x\bullet\ x \in FChainU\ f\ r \Rightarrow p\ x)$

fchainu_induction_thm4

$\vdash \forall\ r\ f\ p$
$\bullet\ CcRpoU\ r$
$\wedge\ Increasing\ r\ r\ f$
$\wedge\ (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\wedge\ (\forall\ y\bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \Rightarrow p\ y)$
$\Rightarrow p\ (f\ x))$
$\wedge\ (\forall\ x$
$\bullet\ x \in FChainU\ f\ r$
$\wedge\ x$
$= Lub$
r
$\{y|y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y\}$
$\wedge\ (\forall\ y$
$\bullet\ y \in FChainU\ f\ r \wedge r\ y\ x \wedge \neg\ r\ x\ y$
$\Rightarrow p\ y)$
$\Rightarrow p\ x)$
$\Rightarrow (\forall\ x\bullet\ x \in FChainU\ f\ r \Rightarrow p\ x)$

refl_inverse_lemma

$\vdash \forall\ X\ r\bullet\ Refl\ (X,\ RelInv\ r) \Leftrightarrow Refl\ (X,\ r)$

antisym_inverse_lemma

$\vdash \forall\ X\ r\bullet\ Antisym\ (X,\ RelInv\ r) \Leftrightarrow Antisym\ (X,\ r)$

trans_inverse_lemma

$\vdash \forall\ X\ r\bullet\ Trans\ (X,\ RelInv\ r) \Leftrightarrow Trans\ (X,\ r)$

partialorder_inverse_lemma

$\vdash \forall\ X\ r$
$\bullet\ PartialOrder\ (X,\ RelInv\ r) \Leftrightarrow PartialOrder\ (X,\ r)$

trich_inverse_lemma

$\vdash \forall\ X\ r \bullet\ Trich\ (X,\ RelInv\ r) \Leftrightarrow Trich\ (X,\ r)$

linearorder_inverse_lemma

$\vdash \forall\ X\ r \bullet\ LinearOrder\ (X,\ RelInv\ r) \Leftrightarrow LinearOrder\ (X,\ r)$

rpo_inverse_lemma

$\vdash \forall\ X\ r \bullet\ Rpo\ (X,\ RelInv\ r) \Leftrightarrow Rpo\ (X,\ r)$

isub_inverse_lemma

$\vdash \forall\ X\ r\ x \bullet\ IsUb\ (RelInv\ r)\ X\ x \Leftrightarrow IsLb\ r\ X\ x$

islb_inverse_lemma

$\vdash \forall\ X\ r\ x \bullet\ IsLb\ (RelInv\ r)\ X\ x \Leftrightarrow IsUb\ r\ X\ x$

islub_inverse_lemma

$\vdash \forall\ X\ r\ x \bullet\ IsLub\ (RelInv\ r)\ X\ x \Leftrightarrow IsGlb\ r\ X\ x$

isglb_inverse_lemma

$\vdash \forall\ X\ r\ x \bullet\ IsGlb\ (RelInv\ r)\ X\ x \Leftrightarrow IsLub\ r\ X\ x$

islfp_inverse_lemma

$\vdash \forall\ r\ f\ e \bullet\ IsLfp\ (RelInv\ r)\ f\ e \Leftrightarrow IsGfp\ r\ f\ e$

isgfp_inverse_lemma

$\vdash \forall\ r\ f\ e \bullet\ IsGfp\ (RelInv\ r)\ f\ e \Leftrightarrow IsLfp\ r\ f\ e$

increasing_inverse_lemma

$\vdash \forall\ r1\ r2$
$\quad \bullet\ Increasing\ (RelInv\ r1)\ (RelInv\ r2)$
$\qquad = Increasing\ r1\ r2$

chaincocomplete_dual_lemma

$\vdash \forall\ X\ r$
$\quad \bullet\ ChainCoComplete\ (X,\ r)$
$\qquad \Leftrightarrow ChainComplete\ (X,\ RelInv\ r)$

coccrpo_dual_lemma

$\vdash \forall\ X\ r \bullet\ CoCcRpo\ (X,\ r) \Leftrightarrow CcRpo\ (X,\ RelInv\ r)$

coccrpou_dual_lemma

$\vdash \forall\ r \bullet\ CoCcRpoU\ r \Leftrightarrow CcRpoU\ (RelInv\ r)$

fcochainclosed_dual_lemma

$\vdash \forall\ f\ X\ r$
$\quad \bullet\ FCoChainClosed\ f\ (X,\ r)$
$\qquad \Leftrightarrow FChainClosed\ f\ (X,\ RelInv\ r)$

fcochain_dual_lemma

$\vdash \forall\ X\ r\ f \bullet\ FCoChain\ f\ (X,\ r) = FChain\ f\ (X,\ RelInv\ r)$

fcochain_lemma1

$\vdash \forall\ X\ r\ f$
$\quad \bullet\ FCoChainClosed\ f\ (X,\ r) \Rightarrow FCoChain\ f\ (X,\ r) \subseteq X$

fcochain_fcochainclosed_lemma

$\vdash \forall\ X\ r\ f$
$\quad \bullet\ Antisym\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\qquad \Rightarrow FCoChainClosed\ f\ (FCoChain\ f\ (X,\ r),\ r)$

fcochain_fcochainclosed_lemma2

$\vdash \forall\ X\ r\ f$
$\quad \bullet\ Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\qquad \Rightarrow FCoChainClosed\ f\ (FCoChain\ f\ (X,\ r),\ r)$

fcoc_fcocclosed_fcoc_lemma

$\vdash \forall\ X\ r\ f$
$\quad \bullet\ Rpo\ (Universe,\ r) \wedge FCoChainClosed\ f\ (X,\ r)$
$\qquad \Rightarrow (\forall\ x$
$\qquad \bullet\ x \in FCoChain\ f\ (X,\ r)$

$$\Rightarrow f\ x \in FCoChain\ f\ (X,\ r))$$

fcochain_lemma2

$$\vdash \forall\ X\ \ Y\ r\ f$$
$$\bullet\ Rpo\ (Universe,\ r)$$
$$\wedge\ FCoChainClosed\ f\ (X,\ r)$$
$$\wedge\ Y \subseteq FCoChain\ f\ (X,\ r)$$
$$\wedge\ FCoChainClosed\ f\ (Y,\ r)$$
$$\Rightarrow\ Y = FCoChain\ f\ (X,\ r)$$

coccrpo_fixp_lemma4

$$\vdash \forall\ X\ r\ f$$
$$\bullet\ Increasing\ r\ r\ f$$
$$\wedge\ Rpo\ (Universe,\ r)$$
$$\wedge\ FCoChainClosed\ f\ (X,\ r)$$
$$\Rightarrow (\forall\ z$$
$$\bullet\ z \in FCoChain\ f\ (X,\ r) \Rightarrow r\ (f\ z)\ z \vee z = f\ z)$$

crpou_ccrpou_lemma

$$\vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow CcRpoU\ r$$

crpou_lub_glb_∅_lemma

$$\vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ x\bullet\ r\ (Lub\ r\ \{\})\ x \wedge r\ x\ (Glb\ r\ \{\}))$$

crpou_fc_clauses

$$\vdash \forall\ r$$
$$\bullet\ CRpoU\ r$$
$$\Rightarrow\ GlbsExist\ r$$
$$\wedge\ LubsExist\ r$$
$$\wedge (\forall\ x\bullet\ r\ x\ x)$$
$$\wedge (\forall\ x\ y\ z\bullet\ r\ x\ y \wedge r\ y\ z \Rightarrow r\ x\ z)$$
$$\wedge (\forall\ x\ y\bullet\ r\ x\ y \wedge r\ y\ x \Rightarrow x = y)$$

crpou_glb_lfp_lemma1

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f$$
$$\Rightarrow (\exists\ e\bullet\ IsGlb\ r\ \{x|r\ (f\ x)\ x\}\ e \wedge IsLfp\ r\ f\ e)$$

crpou_increasing_lfp_lemma1

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ l\bullet\ IsLfp\ r\ f\ l)$$

crpou_increasing_lfp_lemma2

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow IsLfp\ r\ f\ (Lfp_c\ r\ f)$$

crpou_lub_gfp_lemma1

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f$$
$$\Rightarrow (\exists\ e\bullet\ IsLub\ r\ \{x|r\ x\ (f\ x)\}\ e \wedge IsGfp\ r\ f\ e)$$

crpou_increasing_gfp_lemma1

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow (\exists\ g\bullet\ IsGfp\ r\ f\ g)$$

crpou_increasing_gfp_lemma2

$$\vdash \forall\ r\ f$$
$$\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow IsGfp\ r\ f\ (Gfp_c\ r\ f)$$

isglb_glb_crpou_lemma

$$\vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ X\ z\bullet\ IsGlb\ r\ X\ z \Rightarrow Glb\ r\ X = z)$$

islub_lub_crpou_lemma

$$\vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow (\forall\ X\ z\bullet\ IsLub\ r\ X\ z \Rightarrow Lub\ r\ X = z)$$

crpou_fixp_induction_thm

$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge (\forall\ s \bullet (\forall\ x \bullet\ x \in s \Rightarrow p\ x) \Rightarrow p\ (Lub\ r\ s)) \Rightarrow p\ (Lfp_c\ r\ f)$$

crpou_fixp_induction_thm2

$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge (\forall\ s \bullet\ LinearOrder\ (s, r) \wedge (\forall\ x \bullet\ x \in s \Rightarrow p\ x) \Rightarrow p\ (Lub\ r\ s)) \Rightarrow p\ (Lfp_c\ r\ f)$$

contprop_induction_thm

$$\vdash \forall\ r\ f\ p \bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \wedge (\forall\ x \bullet\ p\ x \Rightarrow p\ (f\ x)) \wedge ContProp\ r\ p \Rightarrow p\ (Lfp_c\ r\ f)$$

F2MF_Monotonic_thm

$$\vdash \forall\ f \bullet\ Monotonic\ (Fun2MonoFun\ f)$$

ClosedUnderFun_thm

$$\vdash \forall\ f\ s \bullet\ s\ ClosedUnderFun\ f \Leftrightarrow (\forall\ t \bullet\ t \subseteq s \Rightarrow f\ t \subseteq s)$$

OpenUnderFun_thm1

$$\vdash \forall\ f\ s \bullet\ s\ OpenUnderFun\ f \Leftrightarrow (\forall\ t \bullet\ s \subseteq t \Rightarrow (\forall\ e \bullet\ e \in s \Rightarrow (\exists\ t' \bullet\ t' \subseteq t \wedge e \in f\ t')))$$

OpenUnderFun_thm2

$$\vdash \forall\ f\ s \bullet\ s\ OpenUnderFun\ f \Leftrightarrow (\forall\ t \bullet\ s \subseteq t \Rightarrow s \subseteq Fun2MonoFun\ f\ t)$$

HeredFun_Closed_thm

$$\vdash \forall\ f \bullet\ HeredFun\ f\ \ ClosedUnderFun\ f$$

HeredFun_Closed_thm1

$$\vdash \forall\ f\ x \bullet (\exists\ t \bullet (\forall\ x \bullet\ x \in t \Rightarrow x \in HeredFun\ f) \wedge x \in f\ t) \Rightarrow x \in HeredFun\ f$$

HeredFun_Open_thm

$$\vdash \forall\ f \bullet\ HeredFun\ f\ \ OpenUnderFun\ f$$

HeredFun_induction_thm

$$\vdash \forall\ f\ s \bullet\ s\ ClosedUnderFun\ f \Rightarrow HeredFun\ f \subseteq s$$

HeredFun_induction_thm1

$$\vdash \forall\ f\ s \bullet (\forall\ t \bullet\ t \subseteq s \Rightarrow f\ t \subseteq s) \Rightarrow HeredFun\ f \subseteq s$$

HeredFun_Open_thm1

$$\vdash \forall\ f\ x$$

$\bullet\ x \in \mathit{HeredFun}\ f$
$\Rightarrow (\exists\ t \bullet (\forall\ x \bullet\ x \in t \Rightarrow x \in \mathit{HeredFun}\ f) \wedge x \in f\ t)$

CoHeredFun_Closed_thm
$\vdash \forall\ f \bullet\ \mathit{CoHeredFun}\ f\ \ \mathit{ClosedUnderFun}\ f$

CoHeredFun_Open_thm
$\vdash \forall\ f \bullet\ \mathit{CoHeredFun}\ f\ \ \mathit{OpenUnderFun}\ f$

CoHeredFun_coinduction_thm
$\vdash \forall\ s\ f \bullet\ s\ \mathit{OpenUnderFun}\ f \Rightarrow s \subseteq \mathit{CoHeredFun}\ f$

CoHeredFun_coinduction_thm2
$\vdash \forall\ s\ f \bullet\ s \subseteq \mathit{Fun2MonoFun}\ f\ s \Rightarrow s \subseteq \mathit{CoHeredFun}\ f$

HeredFun_induct_thm0
$\vdash \forall\ f\ s$
$\bullet\ (s \cap \mathit{HeredFun}\ f)\ \mathit{ClosedUnderFun}\ f \Rightarrow \mathit{HeredFun}\ f \subseteq s$

HeredRel_Closed_thm
$\vdash \forall\ f \bullet\ \mathit{HeredRel}\ f\ \ \mathit{ClosedUnderRel}\ f$

HeredRel_Open_thm
$\vdash \forall\ f \bullet\ \mathit{HeredRel}\ f\ \ \mathit{OpenUnderRel}\ f$

HeredRel_induction_thm
$\vdash \forall\ s\ f \bullet\ s\ \mathit{ClosedUnderRel}\ f \Rightarrow \mathit{HeredRel}\ f \subseteq s$

CoHeredRel_Closed_thm
$\vdash \forall\ f \bullet\ \mathit{CoHeredRel}\ f\ \ \mathit{ClosedUnderRel}\ f$

CoHeredRel_Open_thm
$\vdash \forall\ f \bullet\ \mathit{CoHeredRel}\ f\ \ \mathit{OpenUnderRel}\ f$

CoHeredRel_coinduction_thm
$\vdash \forall\ s\ f \bullet\ s\ \mathit{OpenUnderRel}\ f \Rightarrow s \subseteq \mathit{CoHeredRel}\ f$

Hereditary_Closed_thm
$\vdash \forall\ (X, \$<<)\ p$
$\bullet\ \mathit{ClosedUnderProp}\ (X, \$<<)\ (\mathit{Hereditary}\ (X, \$<<)\ p)\ p$

Hereditary_Open_thm
$\vdash \forall\ (X, \$<<)\ p$
$\bullet\ \mathit{OpenUnderProp}\ (X, \$<<)\ (\mathit{Hereditary}\ (X, \$<<)\ p)\ p$

Hereditary_induction_thm
$\vdash \forall\ (X, \$<<)\ s\ p$
$\bullet\ \mathit{ClosedUnderProp}\ (X, \$<<)\ s\ p$
$\Rightarrow \mathit{Hereditary}\ (X, \$<<)\ p \subseteq s$

CoHereditary_Closed_thm
$\vdash \forall\ (X, \$<<)\ p$
$\bullet\ \mathit{ClosedUnderProp}$
$(X, \$<<)$
$(\mathit{CoHereditary}\ (X, \$<<)\ p)$
p

CoHereditary_Open_thm
$\vdash \forall\ (X, \$<<)\ p$
$\bullet\ \mathit{OpenUnderProp}\ (X, \$<<)\ (\mathit{CoHereditary}\ (X, \$<<)\ p)\ p$

CoHereditary_coinduction_thm
$\vdash \forall\ (X, \$<<)\ s\ p$
$\bullet\ \mathit{OpenUnderProp}\ (X, \$<<)\ s\ p$
$\Rightarrow s \subseteq \mathit{CoHereditary}\ (X, \$<<)\ p$

p_induct_thm1
$\vdash \forall\ X'\ R$
$\bullet\ (\forall\ x$

$\bullet\ (\exists\ y\bullet\ R\ y\ x \wedge (\forall\ z\bullet\ R\ z\ y \Rightarrow z \in X')) \Rightarrow x \in X')$
$\Rightarrow I\_closure\ R \subseteq X'$

p_induct_thm2

$\vdash \forall\ X'\ R$
$\bullet\ (\forall\ x$
$\bullet\ (\forall\ y\bullet\ R\ y\ x \Rightarrow (\exists\ z\bullet\ R\ z\ y \wedge z \in X')) \Rightarrow x \in X')$
$\Rightarrow II\_closure\ R \subseteq X'$

I_closed_closure_lemma

$\vdash \forall\ R\bullet\ I\_closed\ R\ (I\_closure\ R)$

II_closed_closure_lemma

$\vdash \forall\ R\bullet\ II\_closed\ R\ (II\_closure\ R)$

I_closed_⋂_lemma

$\vdash \forall\ R\ y\bullet\ (\forall\ x\bullet\ x \in y \Rightarrow I\_closed\ R\ x) \Rightarrow I\_closed\ R\ (\bigcap\ y)$

II_closed_⋂_lemma

$\vdash \forall\ R\ y$
$\bullet\ (\forall\ x\bullet\ x \in y \Rightarrow II\_closed\ R\ x) \Rightarrow II\_closed\ R\ (\bigcap\ y)$

P_closed_⇒_P_thm

$\vdash \forall\ c\ d\ P\ Q\ x\bullet\ x \in P\_closure\ (c,\ d,\ P,\ Q) \Rightarrow P\ x$

Q_closed_⇒_Q_thm

$\vdash \forall\ c\ d\ P\ Q\ x$
$\bullet\ x \in Q\_closure\ (c,\ d,\ P,\ Q) \Rightarrow Q\ x$

15 INDEX