Infinitarily Definable Non-Well-Founded Sets

Roger Bishop Jones

Abstract

This paper is my second approach to set theory conceived as a maximal consistent theory of set comprehension. The principle innovation in this version is to simplify the syntax by removing comprehension, so that the syntactic category of term is no longer required.

Created: 2006/11/29

Last Change Date: 2012/08/11 21:01:52

http://www.rbjones.com/rbjpub/pp/doc/t021.pdf

Id: t024.doc,v 1.19 2012/08/11 21:01:52 rbj Exp

Contents

To Do

-
-

References

[1] Roger Bishop Jones. Set Theory as Consistent Infinitary Comprehension. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t021.pdf`.

1 INTRODUCTION

See t021 for previous discussion. I will put something better here if it works out.

2 INFINITARY LOGIC

SML

```
open_theory "misc2";
force_new_theory "ifol";
force_new_pc "'ifol";
merge_pcs ["'savedthm_cs_∃_proof"] "'ifol";
set_merge_pcs ["hol1", "'GS1", "'misc1", "'misc2", "'ifol"];
```

2.1 Syntax

2.1.1 Constructors, Discriminators and Destructors

Preliminary to presenting the inductive definition of the required classes we define the nuts and bolts operations on the required syntactic entities (some of which will be used in the inductive definition).

A constructor puts together some syntactic entity from its constituents, discriminators distinguist between the different kinds of entity and destructors take them apart.

"Atomic" formulae consist of a relation name together with an indexed collection of arguments. The relation name may be any set. The indexed set of arguments is any set which is a function, i.e. a many-one relation represented as a set of (Wiener-Kuratovski) ordered pairs. The distinction between atomic and compound formulae is made by tagging the former with the ordinal zero and the latter with the ordinal 1, a tagged value in this case being an ordered pair of which the left element is the tag and the right element is the value.

HOL Constant

$$\boldsymbol{MkAf} : GS \times GS \to GS$$

$$\forall lr \bullet MkAf\ lr = (Nat_g\ 0) \mapsto_g ((Fst\ lr) \mapsto_g (Snd\ lr))$$

HOL Constant

$$\boldsymbol{IsAf} : GS \to BOOL$$

$$\forall t \bullet IsAf\ t = fst\ t = (Nat_g\ 0)$$

HOL Constant

$$\boldsymbol{AfRel} : GS \to GS$$

$$AfRel = \lambda x \bullet fst(snd\ x)$$

HOL Constant

$$\boldsymbol{AfPars} : GS \to GS$$

$$AfPars = \lambda x \bullet snd(snd\ x)$$

HOL Constant

$$\boldsymbol{MkCf} : GS \times GS \to GS$$

$$\forall vc \bullet MkCf\ vc = (Nat_g\ 1) \mapsto_g ((Fst\ vc) \mapsto_g (Snd\ vc))$$

HOL Constant

$$\boldsymbol{IsCf} : GS \to BOOL$$

$$\forall t \bullet IsCf\ t = fst\ t = (Nat_g\ 1)$$

HOL Constant

$$\boldsymbol{CfVars} : GS \to GS$$

$$CfVars = \lambda x \bullet fst(snd\ x)$$

HOL Constant

$$\boldsymbol{CfForms} : GS \to GS$$

$$CfForms = \lambda x \bullet snd(snd\ x)$$

$$\boldsymbol{Is\_clauses} =$$
$$\vdash (\forall\ x \bullet IsAf\ (MkAf\ x))$$
$$\wedge (\forall\ x \bullet \neg\ IsAf\ (MkCf\ x))$$
$$\wedge (\forall\ x \bullet \neg\ IsCf\ (MkAf\ x))$$
$$\wedge (\forall\ x \bullet IsCf\ (MkCf\ x))$$

$$\boldsymbol{Is\_not\_fc\_clauses} =$$
$$\vdash (\forall\ x \bullet IsAf\ x \Rightarrow \neg\ IsCf\ x) \wedge (\forall\ x \bullet IsCf\ x \Rightarrow \neg\ IsAf\ x)$$

Some derived syntax:

HOL Constant

$$\boldsymbol{MkNot} : GS \to GS$$

$$\forall f \bullet MkNot\ f = MkCf\ (\varnothing_g,\ Pair_g\ f\ f)$$

2.1.2 The Inductive Definition of Syntax

This is accomplished by defining the required closure condition (closure under the above constructors for arguments of the right kind) and then taking the intersection of all sets which satisfy the closure condition.

The closure condition is:

HOL Constant

$$\textbf{\textit{RepClosed}}: GS\ SET \to BOOL$$

$$\forall\ s\bullet\ RepClosed\ s \Leftrightarrow$$
$$(\forall\ n\ is\bullet\ fun\ is \Rightarrow MkAf\ (n,\ is) \in s)$$
$$\wedge\quad (\forall\ vars\ fs\bullet\ X_g\ fs \subseteq s \Rightarrow MkCf\ (vars,\ fs) \in s)$$

The well-formed syntax is then the smallest set closed under these constructions.

HOL Constant

$$\textbf{\textit{Syntax}} : GS\ SET$$

$$Syntax = \bigcap\{x \mid RepClosed\ x\}$$

$$\textbf{\textit{syntax\_⊆\_repclosed\_thm}} =$$
$$\vdash \forall\ s\bullet\ RepClosed\ s \Rightarrow Syntax \subseteq s$$

This is an "inductive datatype" so we should expect the usual kinds of theorems.

Informally these should say:

- Syntax is closed under the two constructors.
- The syntax constructors are injections, have disjoint ranges, and partition the syntax.
- Any syntactic property which is preserved by the constructors (i.e. is true of any construction if it is true of all its syntactic constituents) is true of everything in syntax (this is an induction principle).

$$\textbf{\textit{repclosed\_syntax\_lemma}} =$$
$$\vdash RepClosed\ Syntax$$

$$\textbf{\textit{repclosed\_syntax\_thm}} =$$
$$\vdash (\forall\ n\ is\bullet\ fun\ is \Rightarrow MkAf\ (n,\ is) \in Syntax)$$
$$\wedge\ (\forall\ vars\ fs$$
$$\bullet\ (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in Syntax) \Rightarrow MkCf\ (vars,\ fs) \in Syntax)$$

$$\textbf{\textit{repclosed\_syntax\_lemma1}} =$$
$$\vdash \forall\ s\bullet\ RepClosed\ s \Rightarrow Syntax \subseteq s$$

$$\textbf{\textit{repclosed\_syntax\_lemma2}} =$$
$$\vdash \forall\ p\bullet\ RepClosed\ \{x|p\ x\} \Rightarrow (\forall\ x\bullet\ x \in Syntax \Rightarrow p\ x)$$

We need to be able to define functions by recursion over this syntax. To do that we need to prove that the syntax of comprehensions is well-founded. This is itself equivalent to an induction principle, so we can try and derive it using the induction principles already available.

We must first define the relation of priority over the syntax, i.e. the relation between an element of the syntax and its constitutents.

HOL Constant

$$\boldsymbol{ScPrec} : GS\ REL$$

$$\forall \alpha\ \gamma \bullet\ ScPrec\ \alpha\ \gamma \Leftrightarrow$$
$$\quad \exists ord\ fs \bullet\ \alpha \in_g fs \wedge \{\alpha;\ \gamma\} \subseteq Syntax \wedge \gamma = MkCf\ (ord,\ fs)$$

$$\boldsymbol{ScPrec\_tc\_\in\_thm} =$$
$$\quad \vdash \forall\ x\ y \bullet\ ScPrec\ x\ y \Rightarrow tc\ \$\in_g\ x\ y$$

$$\boldsymbol{well\_founded\_ScPrec\_thm} =$$
$$\quad \vdash well\_founded\ ScPrec$$

$$\boldsymbol{well\_founded\_tcScPrec\_thm} =$$
$$\quad \vdash well\_founded\ (tc\ ScPrec)$$

SML

$$val\ \boldsymbol{SC\_INDUCTION\_T} = WFCV\_INDUCTION\_T\ well\_founded\_ScPrec\_thm;$$
$$val\ \boldsymbol{sc\_induction\_tac} = wfcv\_induction\_tac\ well\_founded\_ScPrec\_thm;$$

The set Syntax gives us the syntactically well-formed phrases of our language. It will be useful to have some predicates which incorporate well-formedness, which are defined here.

$$\boldsymbol{syntax\_disj\_thm} =$$
$$\quad \vdash \forall\ x$$
$$\quad\quad \bullet\ x \in Syntax$$
$$\quad\quad\quad \Rightarrow (\exists\ r\ pars \bullet\ fun\ pars \wedge x = MkAf\ (r,\ pars))$$
$$\quad\quad\quad\quad \vee (\exists\ vars\ fs \bullet (\forall\ y \bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs))$$

$$\boldsymbol{syntax\_cases\_thm} =$$
$$\quad \vdash \forall\ x \bullet\ x \in Syntax \Rightarrow IsAf\ x \vee IsCf\ x$$

$$\boldsymbol{is\_fc\_clauses} =$$
$$\quad \vdash \forall\ x$$
$$\quad\quad \bullet\ x \in Syntax$$
$$\quad\quad\quad \Rightarrow (IsAf\ x \Rightarrow (\exists\ r\ pars \bullet\ fun\ pars \wedge x = MkAf\ (r,\ pars)))$$
$$\quad\quad\quad\quad \wedge (IsCf\ x$$
$$\quad\quad\quad\quad\quad \Rightarrow (\exists\ vars\ fs$$
$$\quad\quad\quad\quad\quad \bullet (\forall\ y \bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs)))$$

$$\boldsymbol{syn\_proj\_clauses} =$$
$$\quad \vdash (\forall\ l\ r \bullet\ AfRel\ (MkAf\ (l,\ r)) = l)$$
$$\quad\quad \wedge (\forall\ l\ r \bullet\ AfPars\ (MkAf\ (l,\ r)) = r)$$
$$\quad\quad \wedge (\forall\ v\ f \bullet\ CfVars\ (MkCf\ (v,\ f)) = v)$$
$$\quad\quad \wedge (\forall\ v\ f \bullet\ CfForms\ (MkCf\ (v,\ f)) = f)$$

$$\textbf{\textit{is\_fc\_clauses2}} =$$
$$\vdash \forall\ x\bullet\ x \in \mathit{Syntax} \Rightarrow \mathit{IsCf}\ x \Rightarrow (\forall\ y\bullet\ y \in_g \mathit{CfForms}\ x \Rightarrow y \in \mathit{Syntax})$$

$$\textbf{\textit{stn\_con\_neq\_clauses}} =$$
$$\vdash \forall\ x\ y\bullet \neg\ \mathit{MkAf}\ x = \mathit{MkCf}\ y$$

$$\textbf{\textit{syn\_comp\_fc\_clauses}} =$$
$$\vdash \forall\ v\ f\bullet\ \mathit{MkCf}\ (v, f) \in \mathit{Syntax} \Rightarrow (\forall\ y\bullet\ y \in_g f \Rightarrow y \in \mathit{Syntax})$$

$$\textbf{\textit{scprec\_fc\_clauses}} =$$
$$\vdash \forall\ \alpha\ \gamma\ \mathit{vars}\ \mathit{fs}\bullet\ \gamma \in \mathit{Syntax} \Rightarrow \gamma = \mathit{MkCf}\ (\mathit{vars}, \mathit{fs}) \wedge \alpha \in_g \mathit{fs} \Rightarrow \mathit{ScPrec}\ \alpha\ \gamma$$

$$\textbf{\textit{scprec\_fc\_clauses2}} =$$
$$\vdash \forall\ t\bullet\ t \in \mathit{Syntax} \Rightarrow \mathit{IsCf}\ t \Rightarrow (\forall\ f\bullet\ f \in_g \mathit{CfForms}\ t \Rightarrow \mathit{ScPrec}\ f\ t)$$

2.2 Semantics

The semantics of infinitary first order logic is given by defining "truth in an interpretation".

2.2.1 Domains

We consider here some of the value domains which are significant in the semantics.

The following type abbreviations are introduced:

RV Relation Value - the arguments to a relation can be represented by indexed sets (think of the indices as parameter names), and a relation is then a truth valued function over these indexed sets (a set of indexed sets won't do because we have three truth values). Note that relations need not have a definite arity, and the function representing a relation must be total over the entire type of indexed sets. There are questions about how best ordinary n-ary relations should be represented, one obvious choice would be to make the truth value undefined for any indexed sets which do not have exactly the right number of numerical indices.

ST Structure = a structure is a domain of disccourse (a set) together with an indexed set of relations over that domain. Ordinals are used for relation names as well as for variable names (no ambiguity arises) and a collection of relations can therefore be modelled in the same way as a relation valued variable assignment.

SML
```
declare_type_abbrev("RV", ["'a","'b"], ⌜:'a IS → 'b⌝);
declare_type_abbrev("ST", ["'a","'b"], ⌜:'a SET × ('a, 'b) RV IS⌝);
```

To help in the location of fixed points we want a semantics which is monotonic, and therefore define here orderings on these domains relative to which we expect the sematantics to be monotonic.

The ordering on relations derives from the ordering on the truth values, using the operator *Pw*.

HOL Constant

$$\textbf{\textit{RvO}} : ('b \to 'b \to BOOL) \to ('a, 'b)\ RV \to ('a, 'b)\ RV \to BOOL$$

$$\forall r \bullet\ RvO\ r = Pw\ r$$

rvo_lubs_exist_thm =

$$\vdash \forall\ r \bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvO\ r)$$

rvo_glbs_exist_thm =

$$\vdash \forall\ r \bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvO\ r)$$

HOL Constant

$$\textbf{\textit{RvIsO}} : ('b \to 'b \to BOOL) \to ('a, 'b)\ RV\ IS \to ('a, 'b)\ RV\ IS \to BOOL$$

$$\forall r \bullet\ RvIsO\ r = IsEO\ (RvO\ r)$$

rviso_lubs_exist_thm =

$$\vdash \forall\ r \bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvIsO\ r)$$

rviso_glbs_exist_thm =

$$\vdash \forall\ r \bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvIsO\ r)$$

HOL Constant

$$\textbf{\textit{StO}} : ('b \to 'b \to BOOL) \to ('a, 'b)\ ST \to ('a, 'b)\ ST \to BOOL$$

$$\forall r \bullet\ StO\ r = DerivedOrder\ Snd\ (IsEO\ (RvO\ r))$$

sto_lubs_exist_thm =

$$\vdash \forall\ r \bullet\ LubsExist\ r \Rightarrow LubsExist\ (StO\ r)$$

sto_glbs_exist_thm =

$$\vdash \forall\ r \bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (StO\ r)$$

2.2.2 Manipulating Valuations

In this syntax, by contrast with that in [1], we do not require variables to be ordinals, they may be arbitrary sets, since no steps are necessary to avoid variable capture. There was no need even in [1], the use of a transfinite version of DeBriujn indices was a hang over from the PolySets where something of the kind really was needed.

This function is used in the evaluation of atomic formulae. Given a set of indices (the names of actual parameters to an atomic formula, which are always variables) and an indexed set (the values of the variables) this function returns an indexed set which contains the values for the actual parameters to the relation.

HOL Constant

$$\textbf{\textit{VarComp}} : GS \to 'a\ IS \to 'a\ IS$$

$$\forall m\ is \bullet\ VarComp\ m\ is =$$
$$\quad \lambda v \bullet\ if\ v \in_g dom\ m\ then\ is\ (m\ _g\ v)\ else\ dpoB$$

2.2.3 Formula Evaluation

Now we define the evaluation of formulae, i.e. the notion of truth in a structure given a variable assignment.

There are two cases in the syntax, atomic and compound formulae. The truth values of the atomic formulae are obtained from an infinitary structure given the values of the arguments, which are always variables, i.e. to evaluate an atomic formula you look up the values of the arguments in the current context (variable assignment) and then look up the truth value of the stipulated relation for those arguments in the structure.

HOL Constant

$$\textbf{\textit{EvalAf}} : 't\ REL \to GS \to ('a, 't)\ ST \to ('a, 't)\ RV$$

$$\forall \$\leq_t\ (at{:}GS)\ (st{:}('a, 't)\ ST)\ (va{:}'a\ IS) \bullet\ EvalAf\ \$\leq_t\ at\ st\ va =$$
$$\quad let\ r = AfRel\ at$$
$$\quad and\ pars = AfPars\ at$$
$$\quad in$$
$$\qquad let\ rv = (Snd\ st)\ r$$
$$\qquad in$$
$$\qquad\quad if\ dpoUdef\ rv$$
$$\qquad\quad then\ Lub\ \$\leq_t\ \{\}$$
$$\qquad\quad else$$
$$\qquad\qquad if\ dpoOdef\ rv$$
$$\qquad\qquad then\ Glb\ \$\leq_t\ \{\}$$
$$\qquad\qquad else\ (dpoV\ rv)\ (VarComp\ pars\ va)$$

To evaluate a compound formula you must first evaluate the constituent formulae in every context obtable by modification of the variables bound by the compound formula. You need only remember the resulting truth values, the compound formulae are in this sense "truth functional", and, though this may involve evaluating a very large number of instances of subformulae, it can only yield some subset of $\{pTrue,\ pFalse,\ pU\}$.

The following definition shows how the truth values of the constituents of a compound formula then determines the truth value of the compound formula.

SML

$$declare\_type\_abbrev(\texttt{"}\textbf{\textit{CFE}}\texttt{"},\ [\texttt{"}'t\texttt{"}],\ \ulcorner{:}'t\ SET \to 't\urcorner);$$

HOL Constant

$$\boldsymbol{EvalCf\_tf3} : TTV\ CFE$$

$$\forall results\bullet\ EvalCf\_tf3\ results =$$
$$\quad if\ results \subseteq \{pTrue\}\ then\ pFalse$$
$$\quad\quad else\ if\ (pFalse) \in results\ then\ pTrue$$
$$\quad\quad else\ pU$$

HOL Constant

$$\boldsymbol{EvalCf\_tf4} : FTV\ CFE$$

$$\forall results\bullet\ EvalCf\_tf4\ results =$$
$$\quad if\ results \subseteq \{fTrue\}\ then\ fFalse$$
$$\quad\quad else\ if\ results \subseteq \{fTrue; fB\}\ then\ fB$$
$$\quad\quad else\ if\ fT \in results\ then\ fT\ else\ fTrue$$

This definition shows how the set of truth values of instances of the constituents is obtained from the denotations of the constituent formulae.

HOL Constant

$$\boldsymbol{EvalCf} : 't\ CFE \to GS \to ('a, 't)\ ST \to ('a, 't)\ RV\ SET \to ('a, 't)\ RV$$

$$\forall etf\ f\bullet\ EvalCf\ etf\ f = \lambda st\ rvs\ va\bullet$$
$$\quad let\ \nu = CfVars\ f$$
$$\quad and\ V = Fst\ st$$
$$\quad in\ etf\ \{pb \mid \exists rv\ v\bullet$$
$$\quad\quad rv \in rvs$$
$$\quad\quad \wedge\ IsDom\ v = X_g\ \nu$$
$$\quad\quad \wedge\ IsRan\ v \subseteq V$$
$$\quad\quad \wedge\ pb = rv\ (IsOverRide\ v\ va)\}$$

Now we define a parameterised functor of which the semantic function is a fixed point.

HOL Constant

$$\boldsymbol{EvalFormFunct} : 't\ CFE \times 't\ REL \times ('a, 't)\ ST \to (GS \to ('a, 't)\ RV) \to (GS \to ('a, 't)\ RV)$$

$$\forall cfe\ \$\leq_t\ st\bullet\ EvalFormFunct\ (cfe, \$\leq_t, st) = \lambda ef\ f\bullet$$
$$\quad if\ f \in Syntax$$
$$\quad then\ if\ IsAf\ f$$
$$\quad\quad then\ EvalAf\ \$\leq_t\ f\ st$$
$$\quad\quad else$$
$$\quad\quad\quad if\ IsCf\ f$$
$$\quad\quad\quad then\ let\ rvs = FunImage\ ef\ (X_g(CfForms\ f))$$
$$\quad\quad\quad\quad in\ \ EvalCf\ cfe\ f\ st\ rvs$$
$$\quad\quad\quad else\ \epsilon x\bullet T$$
$$\quad else\ \epsilon x\bullet T$$

HOL Constant

$$\boldsymbol{EvalForm} : 't\ CFE \times 't\ REL \times ('a, 't)\ ST \to GS \to ('a, 't)\ RV$$

$$\forall cfe\ \$\leq_t\ st\bullet\ EvalForm\ (cfe, \$\leq_t, st) = fix\ (EvalFormFunct\ (cfe, \$\leq_t, st))$$

To use this definition we need to show that there exists a fixed point, for which we must show that the functor respects some well-founded relation.

$\boldsymbol{evalformfunct\_respect\_thm} =$

$\vdash \forall\ (V, r)\bullet\ EvalFormFunct\ (V, r)\ respects\ ScPrec$

$\boldsymbol{evalformfunct\_fixp\_lemma} =$

$\vdash \forall\ st\bullet\ EvalForm\ st = EvalFormFunct\ st\ (EvalForm\ st)$

$\boldsymbol{evalformfunct\_thm} =$

$\vdash \forall\ st$
$\bullet\ EvalForm\ st$
$= (\lambda\ f$
$\bullet\ if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ EvalAf\ f\ st$
$else\ if\ IsCf\ f$
$then$
$let\ rvs = FunImage\ (EvalForm\ st)\ (X_g\ (CfForms\ f))$
$in\ EvalCf\ f\ st\ rvs$
$else\ \epsilon\ x\bullet\ T)$
$else\ \epsilon\ x\bullet\ T)$

$\boldsymbol{evalformfunct\_thm2} =$

$\vdash \forall\ st\ f$
$\bullet\ EvalForm\ st\ f$
$= (if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ EvalAf\ f\ st$
$else\ if\ IsCf\ f$
$then$
$let\ rvs = FunImage\ (EvalForm\ st)\ (X_g\ (CfForms\ f))$
$in\ EvalCf\ f\ st\ rvs$
$else\ \epsilon\ f\bullet\ T)$
$else\ \epsilon\ f\bullet\ T)$

2.2.4 Some Orderings

In order to prove that the semantics is monotonic, we must first define the partial orderings relative to which the semantics is monotonic, and we must obtain fixpoint theorems for the orderings.

We have at present two cases under consideration, according to whether three or four truth values are adopted.

The three valued case turns out in some respects more complex than the four valued case, because it is necessary to make do with chain completeness and the fixed point theorem is more difficult to prove. I will therefore progress only the four valued case until I find a reason to further progress the three valued case.

Here is the beginning of the three valued case which I started before.

It is also necessary to prove that these partial orderings are CCPOs (chain complete partial orders), this being the weakest condition for which we have a suitable fixed point theorem. It is convenient to be slightly more definite, to make the orderings all reflexive, and show that they are reflexive CCPOs (for which we use the term CCRPO).

The following ordering is applicable to partial relations.

SML

$declare\_infix(300,\ "\leq_{ft3}");$

HOL Constant

$$\$\leq_{ft3} : ('a \to TTV) \to ('a \to TTV) \to BOOL$$

$$\$\leq_{ft3} = Pw\ \$\leq_{t3}$$

$$\boldsymbol{ccrpou\_\leq_{ft3}\_thm} =$$
$$\vdash CcRpoU\ \$\leq_{ft3}$$

Lets now get on with the four valued case.

SML

$declare\_infix(300,\ "\leq_{ft4}");$

HOL Constant

$$\$\leq_{ft4} : ('a \to FTV) \to ('a \to FTV) \to BOOL$$

$$\$\leq_{ft4} = Pw\ \$\leq_{t4}$$

$$\boldsymbol{ccrpou\_\leq_{ft4}\_thm} =$$
$$\vdash\ \$\leq_{ft4}$$

$$\boldsymbol{evalcf\_tf4\_increasing\_lemma} =$$
$$\vdash Increasing\ (SetO\ \$\leq_{t4})\ \$\leq_{t4}\ EvalCf\_tf4$$

2.2.5 Monotonicity

$$\vdash \forall\ tr\ g\bullet\ CRpoU\ tr \Rightarrow Increasing\ (StO\ tr)\ (RvO\ tr)\ (EvalAf\ tr\ g)$$

To get a monontonicity result for the semantics of first order logic it is necessary to adjust the type of the semantic function.

The function which we wish to be monotonic is the mappings for each fixed domain of discourse and each particular formula, which take an indexed set of relations (corresponding to some interpretation over the gived domain) and return the relation represented by the formulae in that context.

The following function accepts one compound argument containing the relevent context and yields a function which we expect to be monotonic:

HOL Constant

$$\mathbf{MonoEvalForm} : 't\ CFE \times 't\ REL \times 'a\ SET \times GS \to ('a, 't)\ RV\ IS \to ('a, 't)\ RV$$

$$\forall c\ r\ s\ g\ ris\bullet\ MonoEvalForm\ (c,\ r,\ s,\ g)\ ris = EvalForm\ (c,\ r,\ (s,\ ris))\ g$$

$$\vdash \forall\ c\ r\ s\ g \bullet\ CRpoU\ r \wedge Increasing\ (SetO\ r)\ r\ c \Rightarrow Increasing\ (RvIsO\ r)\ (RvO\ r)\ (MonoEvalForm\ (c,\ r,\ s,\ g))$$

$$\vdash \forall\ c\ r\ s\ g \bullet\ CRpoU\ r \wedge Increasing\ (SetO\ r)\ r\ c \Rightarrow Increasing\ (RvIsO\ r)\ (RvO\ r)\ (\lambda\ ris\bullet\ EvalForm\ (c,\ r,\ s,\ ris)\ g)$$

3 SET THEORY

We now narrow our interest to just one theory, set theory. This will be treated using the above formalisation of infinitary logic, and will be the infinitary language with just two binary relations, equality and membership.

We consider set theory as the theory of extensions. The 'naive' approach to this is the theory with equality and membership which has equality and extensionality axioms and the principle of set comprehension, according to which to every formula with one free variable there is a set whose extension is those elements for which the formula will be true if the free variable denotes that element. This theory unfortunately is inconsistent, but, more than one century after this discovery we still have neither a wholly satisfactory explanation of why this is the case, nor a theory which can be argued to be a maximal consistent weakening of that ontological principle. Of course, there may be no such theory, but it is our purpose here to look further into this matter.

This will be done by looking for maximal subsets of the infinitarily definable properties which can be realised in a consistent set theory. The definition of infinitary first order logic above stipulated a class of properties relative to some give relational structure, and tells us the meaning of these formulae. We will be seeking subsets of the formulae which provide an interpretation of set theory.

This will be done by formulating the semantics of set theory as a functor operating on the relational structure for which the existence of a fixed point determines the required interpretation.

We are seeking a functor which when supplied with membership and equality relations will deliver new relationships at least as detailed as the original (they are partial relationships). This is what we now define.

We define a functor which takes a relational structure containing a membership and an equality relation, over a domain which is some unspecified subset of the formulae of infinitary logic defined above, and computes a new similar structure. The new structure will be that of the sets infinitarily definable in the first structure by formulae in the domain of discourse. This functor may be view as the giving a semantics to the language of infinitary first order set theory, which is does by adding the the semantics of the logic above, an account of the meaning of the membership and equality relations. This account is recursive and is therefore expressed as a functor, and will be well-defined only if the functor has a fixed point. The functor will be monotone and will therefore have a fixed point, but this will be in general a pair of partial relations, and we will be seeking particular subsets of the language for which there is a definite fixed point such that the relations are everywhere either true or false. From such a definite fixed point an interpretation of the classical two-valued set theory may be constructed.

3.1 Packing and Unpacking Relationship Pairs

The format of a structure, as used in the specification of infinitary logic above, supports arbitrary structures as indexed sets. The definitions in this section provide for conversion between indexed sets and pairs of relationships.

```
SML
declare_type_abbrev ("PR", ["'a", "'b"], ⌜:('a, 'b) BR × ('a, 'b) BR⌝);
```

HOL Constant

$$\textbf{\textit{PackBinRel}}: (\prime a, \prime t\ DPO)\ BR \to (\prime a, \prime t\ DPO)\ RV$$

$$\forall\ r\bullet\ PackBinRel\ r = \lambda isp{:}\ \prime a\ IS\bullet$$
$$\quad if\ IsDom\ isp = \{Nat_g\ 0;\ Nat_g\ 1\} \wedge IsOd\ isp = \{\}$$
$$\quad then\ r\ (dpoV\ (isp\ (Nat_g\ 0)))\ (dpoV\ (isp\ (Nat_g\ 1)))$$
$$\quad else\ if\ IsOd\ isp = \{\}\ then\ dpoB\ else\ dpoT$$

For the monotonicity proof it is useful to define the following ordering over binary relations:

HOL Constant

$$\textbf{\textit{BrO}}: (\prime t,\ BOOL)\ BR \to ((\prime a,\prime t)\ BR,\ BOOL)BR$$

$$\forall\ r\bullet\ BrO\ r = Pw\ (Pw\ r)$$

$$\textbf{\textit{packbinrel\_increasing\_lemma}} =$$
$$\vdash \forall\ r\bullet\ Rpo\ (Universe,\ r) \Rightarrow Increasing\ (BrO\ r)\ (RvO\ r)\ PackBinRel$$

HOL Constant

$$\textbf{\textit{UnpackBinRel}}\ :\ (\prime a,\ \prime t)\ RV \to (\prime a,\ \prime t)\ BR$$

$$\forall\ rv\bullet\ UnpackBinRel\ rv = \lambda x\ y\bullet\ rv$$
$$\quad (\lambda p\bullet$$
$$\qquad if\ p = Nat_g\ 0$$
$$\qquad then\ dpoE\ x$$
$$\qquad else$$
$$\qquad\quad if\ p = Nat_g\ 1$$
$$\qquad\quad then\ dpoE\ y$$
$$\qquad\quad else\ dpoB)$$

$$\textbf{\textit{unpackbinrel\_increasing\_lemma}} =$$
$$\vdash \forall\ r\bullet\ Increasing\ (RvO\ r)\ (BrO\ r)\ UnpackBinRel$$

HOL Constant

$$\textbf{\textit{PackRelPair}}\ :\ (\prime a,\ \prime t\ DPO)\ PR \to (\prime a,\ \prime t\ DPO)\ RV\ IS$$

$$\forall rp\bullet\ PackRelPair\ rp =$$
$$\quad let\ (\$=_v,\ \$\in_v) = rp\ in$$
$$\quad \lambda rn\bullet$$
$$\quad if\ rn = Nat_g\ 0\ then\ dpoE\ (PackBinRel\ \$=_v)$$
$$\quad else\ if\ rn = Nat_g\ 1\ then\ dpoE\ (PackBinRel\ \$=_v)$$
$$\quad else\ dpoB$$

Here is the relevant ordering on pairs of binary relations:

HOL Constant

$$\textbf{PbrO}: (\prime t,\ BOOL)BR \to ((\prime a,\ \prime t)\ PR,\ BOOL)\ BR$$

$$\forall\ r\bullet\ PbrO\ r = PrO\ (BrO\ r)\ (BrO\ r)$$

$$\textbf{pbro\_crpou\_thm} =$$
$$\vdash \forall\ r\bullet\ CRpoU\ r \Rightarrow CRpoU\ (PbrO\ r) crpou\_increasing\_lfp\_lemma2 =$$
$$\vdash \forall\ r\ f\bullet\ CRpoU\ r \wedge Increasing\ r\ r\ f \Rightarrow IsLfp\ r\ f\ (Lfp_c\ r\ f)$$

$$\textbf{pbro\_}\leq_{t4}\textbf{\_crpou\_thm} =$$
$$\vdash\ CRpoU\ (PbrO\ \$\leq_{t4})$$

$$\textbf{packrelpair\_increasing\_lemma} =$$
$$\vdash \forall\ r$$
$$\bullet\ Rpo\ (Universe,\ r)$$
$$\Rightarrow Increasing\ (PbrO\ r)\ (RvIsO\ r)\ PackRelPair$$

HOL Constant

$$\textbf{UnpackRelPair}\ :\ (\prime a,\ \prime t\ DPO)\ RV\ IS \to (\prime a,\ \prime t\ DPO)\ PR$$

$$\forall rvis{:}(\prime a,\ \prime t\ DPO)\ RV\ IS\bullet\ UnpackRelPair\ rvis =$$
$$let\ f = (\lambda n\bullet$$
$$if\ dpoOdef\ (rvis\ n)$$
$$then\ (\lambda\ x\ y\bullet\ dpoT)$$
$$else$$
$$if\ dpoUdef\ (rvis\ n)$$
$$then\ (\lambda x\ y\bullet\ dpoB)$$
$$else\ UnpackBinRel\ (dpoV\ (rvis\ n)))$$
$$in\ (f\ (Nat_g\ 0),\ f\ (Nat_g\ 1))$$

$$\textbf{unpackrelpair\_increasing\_lemma} =$$
$$\vdash\ Increasing\ (RvIsO\ Dpo)\ (PbrO\ Dpo)\ UnpackRelPair$$

3.2 Recasting of EvalForm

We now recast *EvalForm* for the special case that the structure relative to which formulae are evaluated is a pair of binary relations.

HOL Constant

$$\textbf{EvalFormPr}\ :\ (\prime t\ DPO)\ CFE \times (\prime t\ DPO)\ REL \times \prime a\ SET \times GS$$
$$\to (\prime a,\ \prime t\ DPO)\ PR \to (\prime a,\ \prime t\ DPO)\ RV$$

$$\forall p\bullet\ EvalFormPr\ p = (MonoEvalForm\ p)\ o\ PackRelPair$$

$$
\begin{array}{l}
evalformpr\_increasing\_thm = \\
\quad \vdash \forall\ c\ r\ s\ g \\
\quad\quad \bullet\ CRpoU\ r \wedge Increasing\ (SetO\ r)\ r\ c \\
\quad\quad\quad \Rightarrow Increasing\ (PbrO\ r)\ (RvO\ r)\ (EvalFormPr\ (c,\ r,\ s,\ g))
\end{array}
$$

3.3 Membership and Equality

HOL Constant

$$
\begin{array}{l}
\boldsymbol{ParamZero} : GS \to GS\ IS \\
\hline
\forall p \bullet\ ParamZero\ p = \lambda x \bullet\ if\ x = Nat_g\ 0\ then\ dpoE\ p\ else\ dpoB
\end{array}
$$

All the above material is generic in the type of truth values. The following material is specific to particular sets of truth values.

SML

$$
\begin{array}{l}
declare\_infix\ (301,\ \texttt{"}\in_{\boldsymbol{v}}\texttt{"}); \\
declare\_infix\ (301,\ \texttt{"}=_{\boldsymbol{v}}\texttt{"});
\end{array}
$$

In the following definitions, we are working with four truth values, which may be thought of as true. false, neither or both. So that the definitions are monotonic, the criteria for truth and falsity are given independently so that it may be possible for both truth values to apply.

HOL Constant

$$
\begin{array}{l}
\boldsymbol{MemRel} : GS\ SET \to (GS,\ FTV)\ PR \to (GS,\ FTV)\ BR \\
\hline
\forall\ V\ (pr\colon (GS,\ FTV)\ PR) \bullet\ MemRel\ V\ pr = \\
\qquad let\ (\$=_v,\ \$\in_v) = pr \\
\qquad in\ \lambda v\ w \bullet \\
\qquad\quad if\ v \in V \wedge w \in V \\
\qquad\quad then\ EvalFormPr\ (EvalCf\_tf4,\ \$\leq_{t4},\ V,\ w)\ pr\ (ParamZero\ v) \\
\qquad\quad else\ fB
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{memrel\_increasing\_lemma} = \\
\quad \vdash \forall\ V \bullet\ Increasing\ (PbrO\ \$\leq_{t4})\ (BrO\ \$\leq_{t4})\ (MemRel\ V) \\
\\
\boldsymbol{memrel\_increasing\_lemma2} = \\
\quad \vdash \forall\ V\ x\ y \bullet\ PbrO\ \$\leq_{t4}\ x\ y \Rightarrow BrO\ \$\leq_{t4}\ (MemRel\ V\ x)\ (MemRel\ V\ y)
\end{array}
$$

HOL Constant

$$\boldsymbol{EqRel} : GS\ SET \to (GS,\ FTV)\ BR \to (GS,\ FTV)\ BR$$

$$\begin{array}{l}
\forall\ V\ \ \$\in_v \bullet\ EqRel\ V\ \$\in_v = (\lambda v\ w \bullet\ Lub\ \$\leq_{t4}\ \{t \mid \\
\quad v = w \wedge t = fTrue \\
\quad \vee\ (\exists x \bullet\ x \in V \wedge (fTrue \leq_{t4}\ x \in_v\ v \wedge fFalse \leq_{t4}\ x \in_v\ w \\
\qquad\qquad \vee\ fFalse \leq_{t4}\ x \in_v\ v \wedge fTrue \leq_{t4}\ x \in_v\ w)) \\
\qquad \wedge\ t = fFalse \\
\quad \vee\ (\forall x \bullet\ (x \in V \Rightarrow fTrue \leq_{t4}\ x \in_v\ v \wedge fTrue \leq_{t4}\ x \in_v\ w \\
\qquad\qquad \vee\ fFalse \leq_{t4}\ x \in_v\ v \wedge fFalse \leq_{t4}\ x \in_v\ w) \\
\qquad \wedge\ t = fTrue) \\
\quad \})
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_increasing_lemma} = \\
\quad \vdash \forall\ V \bullet\ Increasing\ (BrO\ \$\leq_{t4})\ (BrO\ \$\leq_{t4})\ (EqRel\ V)
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_increasing_lemma2} = \\
\quad \vdash \forall\ V\ x\ y \bullet\ BrO\ \$\leq_{t4}\ x\ y \Rightarrow BrO\ \$\leq_{t4}\ (EqRel\ V\ x)\ (EqRel\ V\ y)
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_refl_lemma} = \\
\quad \vdash \forall\ V\ pr\ x \bullet\ x \in V \Rightarrow EqRel\ V\ pr\ x\ x = fTrue
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_sym_lemma} = \\
\quad \vdash \forall\ V\ pr\ x\ y \bullet\ x \in V \wedge y \in V \Rightarrow EqRel\ V\ pr\ x\ y = EqRel\ V\ pr\ y\ x
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_ftrue_lemma} = \\
\quad \vdash \forall\ V\ pr\ x\ y \\
\quad\ \bullet\ x \in V \wedge y \in V \\
\qquad \Rightarrow (EqRel\ V\ pr\ x\ y = fTrue \\
\qquad\ \ \Leftrightarrow x = y \\
\qquad\qquad \vee\ (\forall\ z \\
\qquad\qquad \bullet\ z \in V \\
\qquad\qquad\quad \Rightarrow Snd\ pr\ z\ x = fTrue \wedge Snd\ pr\ z\ y = fTrue \\
\qquad\qquad\qquad \vee\ Snd\ pr\ z\ x = fFalse \wedge Snd\ pr\ z\ y = fFalse))
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel_trans_lemma} = \\
\quad \vdash \forall\ V\ pr\ x\ y\ z \\
\quad\ \bullet\ x \in V \wedge y \in V \wedge z \in V \\
\qquad \Rightarrow EqRel\ V\ pr\ x\ y = fTrue \wedge EqRel\ V\ pr\ y\ z = fTrue \\
\qquad \Rightarrow EqRel\ V\ pr\ x\ z = fTrue
\end{array}$$

$$\begin{array}{l}
\boldsymbol{eqrel2_increasing_lemma} = \\
\quad \vdash \forall\ V \bullet\ Increasing\ (BrO\ \$\leq_{t4})\ (BrO\ \$\leq_{t4})\ (EqRel2\ V)
\end{array}$$

3.4 The Semantic Functor

We now define a monotone functor of which, we hope, total fixed points yeild interpretations of the first order language of set theory.

HOL Constant

$$\boldsymbol{SemanticFunctor} : GS\ SET \to (GS,\ FTV)\ PR \to (GS,\ FTV)\ PR$$

$$\forall\ V \bullet\ SemanticFunctor\ V = \lambda(\$=_v,\ \$\in_v) \bullet\ (EqRel\ V\ \$\in_v,\ MemRel\ V\ (\$=_v,\ \$\in_v))$$

$$\boldsymbol{semanticfunctor\_increasing\_thm} =$$
$$\vdash \forall\ V \bullet\ Increasing\ (PbrO\ \$\leq_{t4})\ (PbrO\ \$\leq_{t4})\ (SemanticFunctor\ V)$$

$$\boldsymbol{sf\_lfp\_lemma1} =$$
$$\vdash \forall\ V \bullet\ IsLfp\ (PbrO\ \$\leq_{t4})\ (SemanticFunctor\ V)\ (Lfp_c\ (PbrO\ \$\leq_{t4})\ (SemanticFunctor\ V))$$

$$\boldsymbol{sf\_gfp\_lemma1} =$$
$$\vdash \forall\ V \bullet\ IsGfp\ (PbrO\ \$\leq_{t4})\ (SemanticFunctor\ V)\ (Gfp_c\ (PbrO\ \$\leq_{t4})\ (SemanticFunctor\ V))$$

4 INFINITARILY DEFINABLE MEMBERSHIP STRUCTURES

We are now able to identify and analyse a large class of interpretations of set theory.

Every fixedpoint of the semantic functor which is total, i.e. in which the equality and membership relations over the domain (the V parameter to the semantic functor) gives only true and false (not underdefined or overdefined) yields a membership structure in the usual sense, and this structure is extensional (i.e. satisfies the axiom of extensionality).

In order to define this class of structures it is necessary to prove some facts about total fixedpoints of the semantic functor. We need to know first that the equality relation in such a fixed point will be an equivalence relation (the equivalence classes will be the elements in the domain of the structure). This enables us to define the class of structures, and our next objective is to prove that they are indeed extensional.

From there we intend to proceed by identifying consistent constraints on the fixed points which effectively place lower bounds on the richness of the ontology and permit the proof of progressively stronger ontological principles, which may be thought of as an axiomatisation of a strong non-well-founded set theory. In this way we seek to approach a maximal theory of comprehension, i.e. of sets construed as the extensions of properties.

First we define the notion of a total fixed point of the semantic functor, and then prove various properties of these fized points.

HOL Constant

$$\boldsymbol{TotalOver} : 'a\ SET \to ('a,\ FTV)BR \to BOOL$$

$$\forall\ V\ r\bullet\ TotalOver\ V\ r \Leftrightarrow \forall x\ y\bullet$$
$$\quad if\ x \in V \wedge y \in V$$
$$\quad then\ r\ x\ y = fTrue \vee r\ x\ y = fFalse$$
$$\quad else\ r\ x\ y = fB$$

HOL Constant

$$\boldsymbol{PrTotalOver} : 'a\ SET \to ('a,\ FTV)PR \to BOOL$$

$$\forall\ V\ pr\bullet\ PrTotalOver\ V\ pr \Leftrightarrow TotalOver\ V\ (Fst\ pr) \wedge TotalOver\ V\ (Snd\ pr)$$

HOL Constant

$$\boldsymbol{SFFixp} : GS\ SET \times (GS,\ FTV)PR \to BOOL$$

$$\forall p\bullet\ SFFixp\ p = SemanticFunctor\ (Fst\ p)\ (Snd\ p) = (Snd\ p)$$

HOL Constant

$$\boldsymbol{BoolRel} : (GS,\ FTV)BR \to (GS,\ BOOL)BR$$

$$\forall r\bullet\ BoolRel\ r = \lambda x\ y\bullet\ r\ x\ y = fTrue$$

(not proven)

$$\boldsymbol{eqrel\_equiv\_lemma} =$$
$$\vdash \forall\ V\ pr\bullet\ TotalOver\ V\ (EqRel\ V\ pr) \Rightarrow Equiv\ (V,\ BoolRel\ (EqRel\ V\ pr))$$

$$\boldsymbol{eqrel\_totalmem\_extensional\_lemma} =$$
$$\vdash \forall\ V\ \$=_v\ \$\in_v$$
$$\bullet\ TotalOver\ V\ \$\in_v$$
$$\Rightarrow (\forall\ x\ y$$
$$\bullet\ x \in V \wedge y \in V$$
$$\Rightarrow EqRel\ V\ (\$=_v,\ \$\in_v)\ x\ y$$
$$= (if\ \forall\ z\bullet\ z \in V \Rightarrow z \in_v x = z \in_v y\ then\ fTrue\ else\ fFalse))$$

HOL Constant

$$\boldsymbol{SFTotalFixp} : GS\ SET \times (GS,\ FTV)PR \to BOOL$$

$$\forall p\bullet\ SFTotalFixp\ p \Leftrightarrow SFFixp\ p \wedge PrTotalOver\ (Fst\ p)\ (Snd\ p)$$

4.0.1 Extensionality of Equality

For our purposes it is helpful to define the following notion of extensionality which applies to functions which operate on four-valued membership relations and yield four-valued relations.

The idea is that a function is extensional if its value for any pair of elements depends only on the extension of the two elements. Since we are working with four truth values, the extensions under consideration are not sets, they are four-valued characteristic functions. However, if the characteristic functions yield the undefined truth value anywhere in the domain under consideration, they are not really knowm to be equal so we require that these characteristic functions are everywhere defined (or overdefined).

HOL Constant

$$\boldsymbol{FtvExtensional} : 'a\ SET \to (('a,\ FTV)\ BR \to ('a,\ FTV)\ BR) \to BOOL$$

$$\forall\ V\ f\bullet\ FtvExtensional\ V\ f = \forall \$\in_v\ x\ y\ z\bullet\ x \in V \wedge y \in V \wedge z \in V$$
$$\wedge\ TotalOver\ V\ \$\in_v \wedge (\forall v\bullet\ v \in V \Rightarrow v \in_v x = v \in_v y)$$
$$\Rightarrow f\ \$\in_v\ x\ z = f\ \$\in_v\ y\ z \wedge f\ \$\in_v\ z\ x = f\ \$\in_v\ z\ y$$

We now show that *EqRel* is in this sense extensional.

$$\boldsymbol{eqrel\_ftvextensional\_lemma} =$$
$$\vdash \forall\ V\ \$=_v\bullet\ FtvExtensional\ V\ (\lambda\ \$\in_v\bullet\ EqRel\ V\ (\$=_v,\ \$\in_v))$$

$$\boldsymbol{totalfixp\_extensional\_lemma} =$$
$$\vdash \forall\ V\ eq\ mem\ \$=_v\ \$\in_v$$
$$\bullet\ SFTotalFixp\ (V,\ eq,\ mem)$$
$$\Rightarrow (let\ \$=_v = BoolRel\ eq\ and\ \$\in_v = BoolRel\ mem$$
$$in\ \forall\ x\ y$$
$$\bullet\ x \in V \wedge y \in V \Rightarrow (x =_v y \Leftrightarrow (\forall\ z\bullet\ z \in V \Rightarrow (z \in_v x \Leftrightarrow z \in_v y))))$$

HOL Constant

$$\boldsymbol{MSfromSFF} : GS\ SET \times (GS,\ FTV)\ PR \to GS\ SET\ SET \times (GS\ SET,\ BOOL)\ BR$$

$$\forall\ (V{:}GS\ SET)\ (pr{:}(GS,\ FTV)\ PR)\bullet\ MSfromSFF\ (V,\ pr) =$$
$$let\ (\$=_v,\ \$\in_v) = (BoolRel\ (Fst\ pr),\ BoolRel\ (Snd\ pr))\ in$$
$$(QuotientSet\ V\ \$=_v,\ \lambda s\ t\bullet\ \forall x\ y\bullet\ x \in s \wedge y \in t \Rightarrow x \in_v y)$$

SML

```
new_parent "membership";
```

4.0.2 Proof Contexts

SML

```
add_pc_thms "'ifol" [];
commit_pc "'ifol";

force_new_pc "ifol";
merge_pcs ["hol", "'GS1", "'misc1", "'misc2", "'ifol"] "ifol";
commit_pc "ifol";

force_new_pc "ifol1";
merge_pcs ["hol1", "'GS1", "'misc1", "'misc2", "'ifol"] "ifol1";
commit_pc "ifol1";
```

5 The Theory ifol

5.1 Parents

membership misc2

5.2 Constants

MkAf	$(GS,\ GS \times GS)\ VA$
IsAf	$(BOOL,\ GS)\ VA$
AfRel	$(GS,\ GS)\ VA$
AfPars	$(GS,\ GS)\ VA$
MkCf	$(GS,\ GS \times GS)\ VA$
IsCf	$(BOOL,\ GS)\ VA$
CfVars	$(GS,\ GS)\ VA$
CfForms	$(GS,\ GS)\ VA$
MkNot	$(GS,\ GS)\ VA$
RepClosed	$(BOOL,\ GS\ \mathbb{P})\ VA$
Syntax	$GS\ \mathbb{P}$
ScPrec	$((BOOL,\ GS)\ VA,\ GS)\ VA$
RvO	$(((BOOL,\ ('a,\ 'b)\ RV)\ VA,\ ('a,\ 'b)\ RV)\ VA,$ $((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$
RvIsO	$((('a,\ 'b)\ RV,\ (('a,\ 'b)\ RV,\ BOOL)\ RV)\ RV,$ $((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$
StO	$(((BOOL,\ ('a,\ 'b)\ ST)\ VA,\ ('a,\ 'b)\ ST)\ VA,$ $((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$
VarComp	$(('a,\ ('a\ DPO,\ GS)\ VA)\ RV,\ GS)\ VA$
EvalAf	$((((('a,\ 't)\ RV,\ ('a,\ 't)\ ST)\ VA,\ GS)\ VA,$ $((BOOL,\ 't)\ VA,\ 't)\ VA)\ VA$
EvalCf_tf3	$TTV\ CFE$
EvalCf_tf4	$FTV\ CFE$
EvalCf	$((((('a,\ 't)\ RV\ CFE,\ ('a,\ 't)\ ST)\ VA,\ GS)\ VA,\ 't\ CFE)\ VA$
EvalFormFunct	$((((('a,\ 't)\ RV,\ GS)\ VA,\ ((('a,\ 't)\ RV,\ GS)\ VA)\ VA,$ $'t\ CFE \times ((BOOL,\ 't)\ VA,\ 't)\ VA \times ('a,\ 't)\ ST)\ VA$
EvalForm	$((('a,\ 't)\ RV,\ GS)\ VA,$ $'t\ CFE \times ((BOOL,\ 't)\ VA,\ 't)\ VA \times ('a,\ 't)\ ST)\ VA$
$\$\leq_{ft3}$	$((BOOL,\ (TTV,\ 'a)\ VA)\ VA,\ (TTV,\ 'a)\ VA)\ VA$
$\$\leq_{ft4}$	$((BOOL,\ (FTV,\ 'a)\ VA)\ VA,\ (FTV,\ 'a)\ VA)\ VA$
MonoEvalForm	$((('a,\ 't)\ RV,\ ('a,\ 't)\ RV)\ RV,$ $'t\ CFE \times ((BOOL,\ 't)\ VA,\ 't)\ VA \times 'a\ \mathbb{P} \times GS)\ VA$
PackBinRel	$(('a,\ 't\ DPO)\ RV,\ (('t\ DPO,\ 'a)\ VA,\ 'a)\ VA)\ VA$
BrO	$(((BOOL,\ (('t,\ 'a)\ VA,\ 'a)\ VA)\ VA,$ $(('t,\ 'a)\ VA,\ 'a)\ VA)\ VA,$ $((BOOL,\ 't)\ VA,\ 't)\ VA)\ VA$
UnpackBinRel	$((('t,\ 'a)\ VA,\ 'a)\ VA,\ ('a,\ 't)\ RV)\ VA$
PackRelPair	$((('a,\ 't\ DPO)\ RV\ DPO,\ GS)\ VA,\ ('a,\ 't\ DPO)\ PR)\ VA$
PbrO	$(((BOOL,\ ('a,\ 't)\ PR)\ VA,\ ('a,\ 't)\ PR)\ VA,$ $((BOOL,\ 't)\ VA,\ 't)\ VA)\ VA$
UnpackRelPair	$(('a,\ 't\ DPO)\ RV,\ ('a,\ 't\ DPO)\ PR)\ RV$

EvalFormPr $(((' a, ' t\ DPO)\ RV, ('a, 't\ DPO)\ PR)\ VA,$
$'t\ DPO\ CFE$
$\times ((BOOL, 't\ DPO)\ VA, 't\ DPO)\ VA$
$\times 'a\ \mathbb{P}$
$\times GS)\ VA$

ParamZero $((GS\ DPO, GS)\ VA, GS)\ VA$

MemRel $((((FTV, GS)\ VA, GS)\ VA, (GS, FTV)\ PR)\ VA, GS\ \mathbb{P})\ VA$

EqRel $((((FTV, GS)\ VA, GS)\ VA, ((FTV, GS)\ VA, GS)\ VA)\ VA,$
$GS\ \mathbb{P})\ VA$

SemanticFunctor
$(((GS, FTV)\ PR, (GS, FTV)\ PR)\ VA, GS\ \mathbb{P})\ VA$

TotalOver $((BOOL, ((FTV, 'a)\ VA, 'a)\ VA)\ VA, 'a\ \mathbb{P})\ VA$

PrTotalOver $((BOOL, ('a, FTV)\ PR)\ VA, 'a\ \mathbb{P})\ VA$

SFFixp $(BOOL, GS\ \mathbb{P} \times (GS, FTV)\ PR)\ VA$

BoolRel $(((BOOL, GS)\ VA, GS)\ VA, ((FTV, GS)\ VA, GS)\ VA)\ VA$

SFTotalFixp $(BOOL, GS\ \mathbb{P} \times (GS, FTV)\ PR)\ VA$

FtvExtensional
$((BOOL,$
$(((FTV, 'a)\ VA, 'a)\ VA, ((FTV, 'a)\ VA, 'a)\ VA)\ VA,$
$'a\ \mathbb{P})\ VA$

MSfromSFF $(GS\ \mathbb{P}\ MS, GS\ \mathbb{P} \times (GS, FTV)\ PR)\ VA$

5.3 Type Abbreviations

$('a, 'b)$ ***RV***	$('a, 'b)\ RV$
$('a, 'b)$ ***ST***	$('a, 'b)\ ST$
$'t$ ***CFE***	$'t\ CFE$
$('a, 'b)$ ***PR***	$('a, 'b)\ PR$

5.4 Fixity

Right Infix 300:
$\leq_{ft3}\ \leq_{ft4}$

Right Infix 301:
$=_v \quad \in_v$

5.5 Definitions

MkAf $\vdash \forall\ lr \bullet\ MkAf\ lr = Nat_g\ 0 \mapsto_g Fst\ lr \mapsto_g Snd\ lr$

IsAf $\vdash \forall\ t \bullet\ IsAf\ t \Leftrightarrow fst\ t = Nat_g\ 0$

AfRel $\vdash AfRel = (\lambda\ x \bullet\ fst\ (snd\ x))$

AfPars $\vdash AfPars = (\lambda\ x \bullet\ snd\ (snd\ x))$

MkCf $\vdash \forall\ vc \bullet\ MkCf\ vc = Nat_g\ 1 \mapsto_g Fst\ vc \mapsto_g Snd\ vc$

IsCf $\vdash \forall\ t \bullet\ IsCf\ t \Leftrightarrow fst\ t = Nat_g\ 1$

CfVars $\vdash CfVars = (\lambda\ x \bullet\ fst\ (snd\ x))$

CfForms $\vdash CfForms = (\lambda\ x \bullet\ snd\ (snd\ x))$

MkNot $\vdash \forall\ f \bullet\ MkNot\ f = MkCf\ (\varnothing_g, Pair_g\ f\ f)$

RepClosed $\vdash \forall\ s$
$\bullet\ RepClosed\ s$

$\Leftrightarrow (\forall\ n\ is \bullet\ fun\ is \Rightarrow MkAf\ (n,\ is) \in s)$
$\quad \wedge (\forall\ vars\ fs \bullet\ X_g\ fs \subseteq s \Rightarrow MkCf\ (vars,\ fs) \in s)$

Syntax $\vdash Syntax = \bigcap \{x | RepClosed\ x\}$

ScPrec $\vdash \forall\ \alpha\ \gamma$
$\bullet\ ScPrec\ \alpha\ \gamma$
$\Leftrightarrow (\exists\ ord\ fs$
$\bullet\ \alpha \in_g fs$
$\wedge \{\alpha;\ \gamma\} \subseteq Syntax$
$\wedge\ \gamma = MkCf\ (ord,\ fs))$

RvO $\vdash \forall\ r \bullet\ RvO\ r = Pw\ r$

RvIsO $\vdash \forall\ r \bullet\ RvIsO\ r = IsEO\ (RvO\ r)$

StO $\vdash \forall\ r \bullet\ StO\ r = DerivedOrder\ Snd\ (IsEO\ (RvO\ r))$

VarComp $\vdash \forall\ m\ is$
$\bullet\ VarComp\ m\ is$
$= (\lambda\ v$
$\bullet\ if\ v \in_g dom\ m\ then\ is\ (m\ _g\ v)\ else\ dpoB)$

EvalAf $\vdash \forall \leq_t\ at\ st\ va$
$\bullet\ EvalAf \leq_t at\ st\ va$
$= (let\ r = AfRel\ at\ and\ pars = AfPars\ at$
$in\ let\ rv = Snd\ st\ r$
$in\ if\ dpoUdef\ rv$
$then\ Lub \leq_t \{\}$
$else\ if\ dpoOdef\ rv$
$then\ Glb \leq_t \{\}$
$else\ dpoV\ rv\ (VarComp\ pars\ va))$

EvalCf_tf3 $\vdash \forall\ results$
$\bullet\ EvalCf\_tf3\ results$
$= (if\ results \subseteq \{pTrue\}$
$then\ pFalse$
$else\ if\ pFalse \in results$
$then\ pTrue$
$else\ pU)$

EvalCf_tf4 $\vdash \forall\ results$
$\bullet\ EvalCf\_tf4\ results$
$= (if\ results \subseteq \{fTrue\}$
$then\ fFalse$
$else\ if\ results \subseteq \{fTrue;\ fB\}$
$then\ fB$
$else\ if\ fT \in results$
$then\ fT$
$else\ fTrue)$

EvalCf $\vdash \forall\ etf\ f$
$\bullet\ EvalCf\ etf\ f$
$= (\lambda\ st\ rvs\ va$
$\bullet\ (let\ \nu = CfVars\ f\ and\ V = Fst\ st$
$in\ etf$
$\{pb$
$|\exists\ rv\ v$
$\bullet\ rv \in rvs$
$\wedge\ IsDom\ v = X_g\ \nu$
$\wedge\ IsRan\ v \subseteq V$

$\wedge\ pb = rv\ (IsOverRide\ v\ va)\}))$

EvalFormFunct

$\vdash \forall\ cfe \leq_t st$
• $EvalFormFunct\ (cfe, \leq_t, st)$
$= (\lambda\ ef\ f$
• $if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ EvalAf \leq_t f\ st$
$else\ if\ IsCf\ f$
$then$
$let\ rvs = FunImage\ ef\ (X_g\ (CfForms\ f))$
$in\ EvalCf\ cfe\ f\ st\ rvs$
$else\ \epsilon\ x\bullet\ T$
$else\ \epsilon\ x\bullet\ T)$

EvalForm $\vdash \forall\ cfe \leq_t st$
• $EvalForm\ (cfe, \leq_t, st)$
$= fix\ (EvalFormFunct\ (cfe, \leq_t, st))$

≤ft3 $\vdash \$\leq_{ft3} = Pw\ \$\leq_{t3}$

≤ft4 $\vdash \$\leq_{ft4} = Pw\ \$\leq_{t4}$

MonoEvalForm $\vdash \forall\ c\ r\ s\ g\ ris$
• $MonoEvalForm\ (c, r, s, g)\ ris$
$= EvalForm\ (c, r, s, ris)\ g$

PackBinRel $\vdash \forall\ r$
• $PackBinRel\ r$
$= (\lambda\ isp$
• if
$IsDom\ isp = \{Nat_g\ 0; Nat_g\ 1\}$
$\wedge\ IsOd\ isp = \{\}$
$then$
r
$(dpoV\ (isp\ (Nat_g\ 0)))$
$(dpoV\ (isp\ (Nat_g\ 1)))$
$else\ if\ IsOd\ isp = \{\}$
$then\ dpoB$
$else\ dpoT)$

BrO $\vdash \forall\ r\bullet\ BrO\ r = Pw\ (Pw\ r)$

UnpackBinRel $\vdash \forall\ rv$
• $UnpackBinRel\ rv$
$= (\lambda\ x\ y$
• rv
$(\lambda\ p$
• $if\ p = Nat_g\ 0$
$then\ dpoE\ x$
$else\ if\ p = Nat_g\ 1$
$then\ dpoE\ y$
$else\ dpoB))$

PackRelPair $\vdash \forall\ rp$
• $PackRelPair\ rp$
$= (let\ (\$=_v, \$\in_v) = rp$
$in\ \lambda\ rn$

$\bullet$ *if* $rn = Nat_g\ 0$
then $dpoE\ (PackBinRel\ \$=_v)$
else if $rn = Nat_g\ 1$
then $dpoE\ (PackBinRel\ \$=_v)$
else $dpoB$)

PbrO $\vdash \forall\ r \bullet\ PbrO\ r = PrO\ (BrO\ r)\ (BrO\ r)$

UnpackRelPair

$\vdash \forall\ rvis$
$\bullet$ $UnpackRelPair\ rvis$
$= (let\ f\ n$
$= (if\ dpoOdef\ (rvis\ n)$
then $\lambda\ x\ y \bullet\ dpoT$
else if $dpoUdef\ (rvis\ n)$
then $\lambda\ x\ y \bullet\ dpoB$
else $UnpackBinRel\ (dpoV\ (rvis\ n)))$
in $(f\ (Nat_g\ 0), f\ (Nat_g\ 1)))$

EvalFormPr $\vdash \forall\ p \bullet\ EvalFormPr\ p = MonoEvalForm\ p\ o\ PackRelPair$

ParamZero $\vdash \forall\ p$
$\bullet$ $ParamZero\ p$
$= (\lambda\ x \bullet\ if\ x = Nat_g\ 0\ then\ dpoE\ p\ else\ dpoB)$

MemRel $\vdash \forall\ V\ pr$
$\bullet$ $MemRel\ V\ pr$
$= (let\ (\$=_v, \$\in_v) = pr$
in $\lambda\ v\ w$
$\bullet$ *if* $v \in V \wedge w \in V$
then
$EvalFormPr$
$(EvalCf\_tf4, \$\leq_{t4}, V, w)$
pr
$(ParamZero\ v)$
else fB)

EqRel $\vdash \forall\ V\ \$\in_v$
$\bullet$ $EqRel\ V\ \$\in_v$
$= (\lambda\ v\ w$
$\bullet$ Lub
$\$\leq_{t4}$
$\{t$
$|v = w \wedge t = fTrue$
$\vee\ (\exists\ x$
$\bullet\ x \in V$
$\wedge\ (fTrue \leq_{t4} x \in_v v$
$\wedge\ fFalse \leq_{t4} x \in_v w$
$\vee\ fFalse \leq_{t4} x \in_v v$
$\wedge\ fTrue \leq_{t4} x \in_v w))$
$\wedge\ t = fFalse$
$\vee\ (\forall\ x$
$\bullet\ (x \in V$
$\Rightarrow fTrue \leq_{t4} x \in_v v$
$\wedge\ fTrue \leq_{t4} x \in_v w$
$\vee\ fFalse \leq_{t4} x \in_v v$
$\wedge\ fFalse \leq_{t4} x \in_v w)$

$$\wedge\ t = fTrue)\})$$

SemanticFunctor

$$\vdash \forall\ V \bullet\ SemanticFunctor\ V = (\lambda\ (\$=_v,\ \$\in_v) \bullet\ (EqRel\ V\ \$\in_v,\ MemRel\ V\ (\$=_v,\ \$\in_v)))$$

TotalOver

$$\vdash \forall\ V\ r \bullet\ TotalOver\ V\ r \Leftrightarrow (\forall\ x\ y \bullet\ if\ x \in V \wedge y \in V\ then\ r\ x\ y = fTrue \vee r\ x\ y = fFalse\ else\ r\ x\ y = fB)$$

PrTotalOver

$$\vdash \forall\ V\ pr \bullet\ PrTotalOver\ V\ pr \Leftrightarrow TotalOver\ V\ (Fst\ pr) \wedge TotalOver\ V\ (Snd\ pr)$$

SFFixp

$$\vdash \forall\ p \bullet\ SFFixp\ p \Leftrightarrow SemanticFunctor\ (Fst\ p)\ (Snd\ p) = Snd\ p$$

BoolRel

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

SFTotalFixp

$$\vdash \forall\ p \bullet\ SFTotalFixp\ p \Leftrightarrow SFFixp\ p \wedge PrTotalOver\ (Fst\ p)\ (Snd\ p)$$

FtvExtensional

$$\vdash \forall\ V\ f \bullet\ FtvExtensional\ V\ f \Leftrightarrow (\forall\ \$\in_v\ x\ y\ z \bullet\ x \in V \wedge y \in V \wedge z \in V \wedge TotalOver\ V\ \$\in_v \wedge (\forall\ v \bullet\ v \in V \Rightarrow v \in_v x = v \in_v y) \Rightarrow f\ \$\in_v\ x\ z = f\ \$\in_v\ y\ z \wedge f\ \$\in_v\ z\ x = f\ \$\in_v\ z\ y)$$

MSfromSFF

$$\vdash \forall\ V\ pr \bullet\ MSfromSFF\ (V,\ pr) = (let\ (\$=_v,\ \$\in_v) = (BoolRel\ (Fst\ pr),\ BoolRel\ (Snd\ pr))\ in\ (V\ /\ \$=_v,\ (\lambda\ s\ t \bullet \forall\ x\ y \bullet\ x \in s \wedge y \in t \Rightarrow x \in_v y)))$$

5.6 Theorems

Is_not_fc_clauses

$$\vdash (\forall\ x \bullet\ IsAf\ x \Rightarrow \neg\ IsCf\ x) \wedge (\forall\ x \bullet\ IsCf\ x \Rightarrow \neg\ IsAf\ x)$$

repclosed_syntax_thm

$$\vdash (\forall\ n\ is \bullet\ fun\ is \Rightarrow MkAf\ (n,\ is) \in Syntax) \wedge (\forall\ vars\ fs \bullet\ (\forall\ x \bullet\ x \in X_g\ fs \Rightarrow x \in Syntax) \Rightarrow MkCf\ (vars,\ fs) \in Syntax)$$

repclosed_syntax_lemma1

$$\vdash \forall\ s \bullet\ RepClosed\ s \Rightarrow Syntax \subseteq s$$

repclosed_syntax_lemma2

$\vdash \forall\ p\bullet\ RepClosed\ \{x|p\ x\} \Rightarrow (\forall\ x\bullet\ x \in Syntax \Rightarrow p\ x)$

well_founded_ScPrec_thm

$\vdash well\_founded\ ScPrec$

well_founded_tcScPrec_thm

$\vdash well\_founded\ (tc\ ScPrec)$

syntax_disj_thm

$$\begin{array}{l}\vdash \forall\ x\\ \quad \bullet\ x \in Syntax\\ \qquad \Rightarrow (\exists\ r\ pars\bullet\ fun\ pars \wedge x = MkAf\ (r,\ pars))\\ \qquad\quad \vee (\exists\ vars\ fs\\ \qquad\quad \bullet (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax)\\ \qquad\qquad \wedge\ x = MkCf\ (vars,\ fs))\end{array}$$

syntax_cases_thm

$\vdash \forall\ x\bullet\ x \in Syntax \Rightarrow IsAf\ x \vee IsCf\ x$

is_fc_clauses

$$\begin{array}{l}\vdash \forall\ x\\ \quad \bullet\ x \in Syntax\\ \qquad \Rightarrow (IsAf\ x\\ \qquad\quad \Rightarrow (\exists\ r\ pars\bullet\ fun\ pars \wedge x = MkAf\ (r,\ pars)))\\ \qquad\quad \wedge\ (IsCf\ x\\ \qquad\quad \Rightarrow (\exists\ vars\ fs\\ \qquad\quad \bullet (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax)\\ \qquad\qquad \wedge\ x = MkCf\ (vars,\ fs)))\end{array}$$

syn_proj_clauses

$$\begin{array}{l}\vdash (\forall\ l\ r\bullet\ AfRel\ (MkAf\ (l,\ r)) = l)\\ \quad \wedge (\forall\ l\ r\bullet\ AfPars\ (MkAf\ (l,\ r)) = r)\\ \quad \wedge (\forall\ v\ f\bullet\ CfVars\ (MkCf\ (v,\ f)) = v)\\ \quad \wedge (\forall\ v\ f\bullet\ CfForms\ (MkCf\ (v,\ f)) = f)\end{array}$$

is_fc_clauses2

$$\begin{array}{l}\vdash \forall\ x\\ \quad \bullet\ x \in Syntax\\ \qquad \Rightarrow IsCf\ x\\ \qquad \Rightarrow (\forall\ y\bullet\ y \in_g CfForms\ x \Rightarrow y \in Syntax)\end{array}$$

syn_con_neq_clauses

$\vdash \forall\ x\ y\bullet \neg\ MkAf\ x = MkCf\ y$

syn_comp_fc_clauses

$$\begin{array}{l}\vdash \forall\ v\ f\\ \quad \bullet\ MkCf\ (v,\ f) \in Syntax \Rightarrow (\forall\ y\bullet\ y \in_g f \Rightarrow y \in Syntax)\end{array}$$

scprec_fc_clauses

$$\begin{array}{l}\vdash \forall\ \alpha\ \gamma\ vars\ fs\\ \quad \bullet\ \gamma \in Syntax\\ \qquad \Rightarrow \gamma = MkCf\ (vars,\ fs) \wedge \alpha \in_g fs\\ \qquad \Rightarrow ScPrec\ \alpha\ \gamma\end{array}$$

scprec_fc_clauses2

$$\begin{array}{l}\vdash \forall\ t\\ \quad \bullet\ t \in Syntax\\ \qquad \Rightarrow IsCf\ t\\ \qquad \Rightarrow (\forall\ f\bullet\ f \in_g CfForms\ t \Rightarrow ScPrec\ f\ t)\end{array}$$

rvo_lubs_exist_thm

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvO\ r)$

rvo_glbs_exist_thm

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvO\ r)$

rviso_lubs_exist_thm

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvIsO\ r)$

rviso_glbs_exist_thm

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvIsO\ r)$

sto_lubs_exist_thm

$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow LubsExist\ (StO\ r)$

sto_glbs_exist_thm

$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (StO\ r)$

evalformfunct_respect_thm

$\vdash \forall\ cfe\ \leq_t\ st$
$\bullet\ EvalFormFunct\ (cfe, \leq_t, st)\ respects\ ScPrec$

evalformfunct_fixp_lemma

$\vdash \forall\ cfe\ \leq_t\ st$
$\bullet\ EvalForm\ (cfe, \leq_t, st)$
$= EvalFormFunct$
$(cfe, \leq_t, st)$
$(EvalForm\ (cfe, \leq_t, st))$

evalformfunct_thm

$\vdash \forall\ cfe\ \leq_t\ st$
$\bullet\ EvalForm\ (cfe, \leq_t, st)$
$= (\lambda\ f$
$\bullet\ if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ EvalAf\ \leq_t\ f\ st$
$else\ if\ IsCf\ f$
$then$
$let\ rvs$
$= FunImage$
$(EvalForm\ (cfe, \leq_t, st))$
$(X_g\ (CfForms\ f))$
$in\ EvalCf\ cfe\ f\ st\ rvs$
$else\ \epsilon\ x\bullet\ T$
$else\ \epsilon\ x\bullet\ T)$

evalformfunct_thm2

$\vdash \forall\ cfe\ \leq_t\ st\ f$
$\bullet\ EvalForm\ (cfe, \leq_t, st)\ f$
$= (if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ EvalAf\ \leq_t\ f\ st$
$else\ if\ IsCf\ f$
$then$
$let\ rvs$
$= FunImage$
$(EvalForm\ (cfe, \leq_t, st))$
$(X_g\ (CfForms\ f))$
$in\ EvalCf\ cfe\ f\ st\ rvs$
$else\ \epsilon\ f\bullet\ T$
$else\ \epsilon\ f\bullet\ T)$

ccrpou_≤$_{ft3}$_thm

$$\vdash \mathit{CcRpoU}\ \$\leq_{ft3}$$

evalform_increasing_thm

$$\begin{array}{l}\vdash \forall\ c\ r\ s\ g \\ \quad \bullet\ \mathit{CRpoU}\ r \wedge \mathit{Increasing}\ (\mathit{SetO}\ r)\ r\ c \\ \qquad \Rightarrow \mathit{Increasing} \\ \qquad\quad (\mathit{RvIsO}\ r) \\ \qquad\quad (\mathit{RvO}\ r) \\ \qquad\quad (\lambda\ \mathit{ris}\bullet\ \mathit{EvalForm}\ (c,\ r,\ s,\ \mathit{ris})\ g)\end{array}$$

pbro_crpou_thm

$$\vdash \forall\ r\bullet\ \mathit{CRpoU}\ r \Rightarrow \mathit{CRpoU}\ (\mathit{PbrO}\ r)$$

pbro_≤$_{t4}$_crpou_thm

$$\vdash \mathit{CRpoU}\ (\mathit{PbrO}\ \$\leq_{t4})$$

evalformpr_increasing_thm

$$\begin{array}{l}\vdash \forall\ c\ r\ s\ g \\ \quad \bullet\ \mathit{CRpoU}\ r \wedge \mathit{Increasing}\ (\mathit{SetO}\ r)\ r\ c \\ \qquad \Rightarrow \mathit{Increasing} \\ \qquad\quad (\mathit{PbrO}\ r) \\ \qquad\quad (\mathit{RvO}\ r) \\ \qquad\quad (\mathit{EvalFormPr}\ (c,\ r,\ s,\ g))\end{array}$$

6 INDEX