Infinitary First Order Set Theory

Roger Bishop Jones

Abstract

The abstract syntax and semantics of an infinitary first order set theory.

Created: 2008/12/05

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

http://www.rbjones.com/rbjpub/pp/doc/t027.pdf

Id: t027.doc,v 1.7 2012/08/11 21:01:52 rbj Exp

Contents

References

[1] Roger Bishop Jones. More Miscellanea. *RBJones.com*, 2010.
`http://www.rbjones.com/rbjpub/pp/doc/t025.pdf`.

1 INTRODUCTION

The idea is to obtain exotic models of set theory using definability in first order set theory as a source of candidate sets.

To ensure that we get all the well-founded sets we start with definability in an infinitary first order set theory (*infos*). Given any membership structure, i.e. a domain of discourse and a membership relation over that domain, each formula of *infos* with one free variable will define a subset of the domain of discourse (you may prefer to think of these as classes, since they will often have the same size as the domain).

If we take as our domain of discourse the formulae of *infos* having a single free variable, then the semantics of *infos* gives rise to a functor which, given one membership relation over that domain will yield another membership relation. Over this domain the functor can have no fixed point, for we would have a formula for the Russell set. To obtain a set theory, we must omit some of these potential sets, i.e. we must consisider subsets as potential domains. Any subset together with a fixed point of the semantic functor will yeild an interpretation of set theory.

It seems clear that some subsets of the formulae of *infos* do have fixed points under the semantics of *infos*. For example, a collection of formulae denoting all the well-founded sets, or those denoting well-founded sets or their complements.

It is the purpose of this work to see whether models for rich non-well-founded set theories can be obtained in this way. The purpose is ultimately to provide a semantic basis for a kind of type theory suitable for "formalised mathematics in the large", in which ordinary mathematics is done primarily using well-founded sets and abstract mathematics often involves non-well-founded sets (theories generic over many types).

SML

```
open_theory "misc2";
force_new_theory "infos";
force_new_pc "'infos";
merge_pcs ["'savedthm_cs_∃_proof"] "'infos";
set_merge_pcs ["misc21", "'infos"];
```

2 SYNTAX

The syntax of infinitary first order set theory is to be encoded as sets in a higher order set theory.

This is (in effect if not in appearance) an "inductive datatype" (albeit transfinite) so we should expect the usual kinds of theorems.

Informally these should say:

1. Syntax is closed under the two constructors.
2. The syntax constructors are injections and have disjoint ranges
3. The ranges of the constructors partition the syntax.
4. 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 the syntax (this is an induction principle).

5. A recursion theorem which supports definition of recursive functions over the syntax.

As well as the constructors, discriminators and destructors are defined.

2.1 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).

Note the terminology here.

constructor A function which constructs a complex structure in the abstract syntax from the immediate constituents of that structure.

discrimiator A propositional function or predicate which tests for a particular class of entities in the abstract syntax. This will normally be that class of entities formed using some particular constructor.

descructor A function which extracts from a complex entity of the abstract syntax one or more of the immediate constutuents from which it was constructed.

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

This is a first order language in which the terms are just variables, and the variables can be arbitrary sets, allowing that there may be very large numbers of them. We therefore proceed directly to the syntax of formulae.

"Atomic" formulae are just membership predicates, of which the operands will always be variables. An atomic formula can therefore be represented in the abstract syntax by an ordered pair, tagged with a zero to distinguish it as am atomic formaula from other kinds of formulae.

The constructor is:

HOL Constant

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

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

A discriminator is:

HOL Constant

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

$$\forall t \bullet IsAf\ t \Leftrightarrow \exists lr \bullet t = MkAf\ lr$$

And we supply two desctructors, the first returning the set in which membership is asserted:

HOL Constant

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

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

and the second returning the set which is alleged to be the member:

HOL Constant

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

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

We have just one way of forming compound formulae, which should be thought of as a generalised infinitary scheffed stroke.

HOL Constant

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

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

The two components of a compound formula are, on the left a set of variables (of any cardinality) and on the right a set of subformulae (also of arbitrary cardinality). Semantically this compound formula should be thought of as an infinitary existentially generalised NOR, and will be true if under any assignment of values to the bound variables at least one of the constituent formulae comes out false. This is complicated somewhet in the formal semantics which follows by the adoption of a four truth values, which however is merely a device to facilitate finding two-valued fixed points in the semantics.

The discriminator is:

HOL Constant

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

$$\forall t\bullet\ IsCf\ \ t \Leftrightarrow \exists vc\bullet\ t = MkCf\ \ vc$$

and the following two functions are the desctructors:

HOL Constant

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

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

HOL Constant

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

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

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

$$
\begin{array}{l}
\textbf{\textit{syn\_proj\_clauses}} = \\
\quad \vdash (\forall\ s\ m\bullet\ AfSet\ (MkAf\ (s,\ m)) = s) \\
\qquad \wedge (\forall\ s\ m\bullet\ AfMem\ (MkAf\ (s,\ m)) = m) \\
\qquad \wedge (\forall\ v\ f\bullet\ CfVars\ (MkCf\ (v,\ f)) = v) \\
\qquad \wedge (\forall\ v\ f\bullet\ CfForms\ (MkCf\ (v,\ f)) = f)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{syn\_con\_inv\_fc\_clauses}} = \\
\quad \vdash \forall\ p \\
\qquad \bullet\ (IsAf\ p \Rightarrow MkAf\ (AfMem\ p,\ AfSet\ p) = p) \\
\qquad\quad \wedge (IsCf\ p \Rightarrow MkCf\ (CfVars\ p,\ CfForms\ p) = p)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{syn\_con\_eq\_clauses}} = \\
\quad \vdash \forall p1\ p2\bullet\ (MkAf\ p1 = MkAf\ p2 \Leftrightarrow p1 = p2) \wedge (MkCf\ p1 = MkCf\ p2 \Leftrightarrow p1 = p2)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{syn\_con\_neq\_clauses}} = \\
\quad \vdash \forall\ x\ y\bullet \neg\ MkAf\ x = MkCf\ y
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{is\_fc\_clauses}1} = \\
\quad \vdash \forall\ x \\
\qquad \bullet\ (IsAf\ x \Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m))) \\
\qquad\quad \wedge (IsCf\ x \Rightarrow (\exists\ vars\ fs\bullet\ x = MkCf\ (vars,\ fs)))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{Is\_not\_cases}} = \\
\quad \vdash \forall\ x\bullet \neg\ IsAf\ x \vee \neg\ IsCf\ x \\
\\
\textbf{\textit{Is\_not\_fc\_clauses}} = \\
\quad \vdash (\forall\ x\bullet\ IsAf\ x \Rightarrow \neg\ IsCf\ x) \wedge (\forall\ x\bullet\ IsCf\ x \Rightarrow \neg\ IsAf\ x)
\end{array}
$$

Some derived syntax:

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{MkNot}} : GS \rightarrow GS \\
\hline
\forall f\bullet\ MkNot\ f = MkCf\ (\varnothing_g,\ Pair_g\ f\ f)
\end{array}
$$

2.2 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

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

$$\forall\ s \bullet\ RepClosed\ s \Leftrightarrow$$
$$(\forall\ s2\ m \bullet\ MkAf\ (s2,\ m) \in s)$$
$$\wedge\quad (\forall\ vars\ fs \bullet\ X_g\ fs \subseteq s \Rightarrow MkCf\ (vars,\ fs) \in s)$$

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

HOL Constant

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

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

I have contrived to arrange the semantics which follows as a monotone function over a complete partial order, so that when I start to look for fixed points of the semantic functor (to give the membership relation for an intepretation of set theory) I can look at the least and greatest fixed points, and either prove one of the total or otherwise employ them to find a total fixed point.

In this scenario the least and greatest fixed points are dual to each other, and there should be a close correspondence between the proofs, the least fixed point corresponding to an inductive and the greatest to a coinductive definition of membership.

When looking at the syntax, which is of course inductively defined, this dualistic mania made me think perhaps that I should also be working with a coinductive version and this provoked my to define *CoSyntax* and replicate some results dual to those proven for the syntax.

I doubt very much that there is any value in this exercise (even if I have got it right), since the pertinent duality flows from the lattice of (four) truth values which I use in the semantics, and this has nothing to do with the syntax. However, pro-tem I will leave the cosyntax definitions in place.

HOL Constant

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

$$\forall\ s \bullet\ RepOpen\ s \Leftrightarrow (\forall\ vars\ fs \bullet\ MkCf\ (vars,\ fs) \in s \Rightarrow X_g\ fs \subseteq s)$$

HOL Constant

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

$$CoSyntax = \bigcup\{x \mid RepOpen\ x\}$$

2.3 Closure

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

$$\textbf{\textit{repopen\_cosyntax\_lemma}} =$$
$$\vdash RepOpen\ CoSyntax$$

repclosed_syntax_thm =

$\vdash (\forall\ s\ m\bullet\ MkAf\ (s,\ m) \in Syntax)$
$\wedge (\forall\ vars\ fs$
$\bullet (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in Syntax) \Rightarrow MkCf\ (vars,\ fs) \in Syntax)$

repopen_cosyntax_thm =

$\vdash \forall\ vars\ fs\bullet\ MkCf\ (vars,\ fs) \in CoSyntax \Rightarrow (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in CoSyntax)$

repclosed_syntax_thm2 =

$\vdash (\forall\ s2\ m\bullet\ MkAf\ (s2,\ m) \in Syntax)$
$\wedge (\forall\ vars\ fs\bullet (\forall\ x\bullet\ x \in_g fs \Rightarrow x \in Syntax) \Rightarrow MkCf\ (vars,\ fs) \in Syntax)$

repopen_cosyntax_thm2 =

$\vdash \forall\ vars\ fs\bullet\ MkCf\ (vars,\ fs) \in CoSyntax \Rightarrow (\forall\ x\bullet\ x \in_g fs \Rightarrow x \in CoSyntax)$

repclosed_syntax_lemma1 =

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

repopen_cosyntax_lemma1 =

$\vdash \forall\ s\bullet\ RepOpen\ s \Rightarrow s \subseteq CoSyntax$

repclosed_syntax_lemma2 =

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

repopen_cosyntax_lemma2 =

$\vdash \forall\ p\bullet\ RepOpen\ \{x|p\ x\} \Rightarrow (\forall\ x\bullet\ p\ x \Rightarrow x \in CoSyntax)$

2.4 Recursion and Induction Principles and Rules

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

ScPrec : $GS\ REL$

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

HOL Constant

ScPrec2 : $GS\ REL$

$\forall\alpha\ \gamma\bullet\ ScPrec2\ \alpha\ \gamma \Leftrightarrow$
$\exists ord\ fs\bullet\ \alpha \in_g fs \wedge \gamma = MkCf\ (ord,\ fs)$

$$\vert\ \textbf{\textit{ScPrec\_tc\_}}\in\textbf{\textit{\_thm}} =$$
$$\vert\quad \vdash \forall\ x\ y\bullet\ ScPrec\ x\ y \Rightarrow tc\ \$\in_g\ x\ y$$
$$\vert$$
$$\vert\ \textbf{\textit{well\_founded\_ScPrec\_thm}} =$$
$$\vert\quad \vdash\ well\_founded\ ScPrec$$

$$\vert\ \textbf{\textit{ScPrec2\_tc\_}}\in\textbf{\textit{\_thm}} =$$
$$\vert\quad \vdash \forall\ x\ y\bullet\ ScPrec2\ x\ y \Rightarrow tc\ \$\in_g\ x\ y$$
$$\vert$$
$$\vert\ \textbf{\textit{well\_founded\_ScPrec2\_thm}} =$$
$$\vert\quad \vdash\ well\_founded\ ScPrec2$$
$$\vert$$
$$\vert\ \textbf{\textit{well\_founded\_tcScPrec2\_thm}} =$$
$$\vert\quad \vdash\ well\_founded\ (tc\ ScPrec2)$$

SML
$$\vert\ val\ \textbf{\textit{SC2\_INDUCTION\_T}} = WFCV\_INDUCTION\_T\ well\_founded\_ScPrec2\_thm;$$
$$\vert\ val\ \textbf{\textit{sc2\_induction\_tac}} = wfcv\_induction\_tac\ well\_founded\_ScPrec2\_thm;$$

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.

$$\vert\ \textbf{\textit{syntax\_disj\_thm}} =$$
$$\vert\quad \vdash \forall\ x$$
$$\vert\quad\quad \bullet\ x \in Syntax$$
$$\vert\quad\quad\quad \Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m))$$
$$\vert\quad\quad\quad\quad \vee (\exists\ vars\ fs\bullet\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs))$$
$$\vert$$
$$\vert\ \textbf{\textit{syntax\_cases\_thm}} =$$
$$\vert\quad \vdash \forall\ x\bullet\ x \in Syntax \Rightarrow IsAf\ x \vee IsCf\ x$$
$$\vert$$
$$\vert\ \textbf{\textit{syntax\_cases\_fc\_clauses}} =$$
$$\vert\quad \vdash \forall\ x\bullet\ x \in Syntax \Rightarrow (\neg\ IsAf\ x \Rightarrow IsCf\ x) \wedge (\neg\ IsCf\ x \Rightarrow IsAf\ x)$$

$$\vert\ \textbf{\textit{is\_fc\_clauses}} =$$
$$\vert\quad \vdash \forall\ x$$
$$\vert\quad\quad \bullet\ x \in Syntax$$
$$\vert\quad\quad\quad \Rightarrow (IsAf\ x \Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m)))$$
$$\vert\quad\quad\quad\quad \wedge (IsCf\ x$$
$$\vert\quad\quad\quad\quad\quad \Rightarrow (\exists\ vars\ fs$$
$$\vert\quad\quad\quad\quad\quad \bullet (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs)))$$
$$\vert$$
$$\vert\ \textbf{\textit{is\_fc\_clauses2}} =$$
$$\vert\quad \vdash \forall\ x\bullet\ x \in Syntax \Rightarrow IsCf\ x \Rightarrow (\forall\ y\bullet\ y \in_g CfForms\ x \Rightarrow y \in Syntax)$$

$\neg \varnothing_g \_\in\_ \boldsymbol{syntax\_lemma} =$
$\vdash \neg\ \varnothing_g \in Syntax$

$\neg \varnothing_g \_\in\_ \boldsymbol{syntax\_lemma2} =$
$\vdash \forall\ x\bullet\ x \in Syntax \Rightarrow \neg\ x = \varnothing_g$

$\neg \varnothing_g \_\in\_ \boldsymbol{syntax\_lemma3} =$
$\vdash \forall\ V\ x\bullet\ x \in V \wedge V \subseteq Syntax \Rightarrow \neg\ x = \varnothing_g$

$\boldsymbol{MkAf\_MkCf\_\neg\_\varnothing_g\_lemma} =$
$\vdash \forall\ x\ y\bullet \neg\ MkAf\ (x,\ y) = \varnothing_g \wedge \neg\ MkCf\ (x,\ y) = \varnothing_g$

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

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

$\boldsymbol{scprec2\_fc\_clauses} =$
$\vdash \forall\ \alpha\ \gamma\ vars\ fs\bullet\ \gamma = MkCf\ (vars,\ fs) \wedge \alpha \in_g fs \Rightarrow ScPrec2\ \alpha\ \gamma$

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

$\boldsymbol{scprec2\_fc\_clauses2} =$
$\vdash \forall\ t\bullet\ IsCf\ t \Rightarrow (\forall\ f\bullet\ f \in_g CfForms\ t \Rightarrow ScPrec2\ f\ t)$

$\boldsymbol{MkAf\_\in\_Syntax\_lemma} =$
$\vdash \forall\ x\ y\bullet\ MkAf\ (x,\ y) \in Syntax$

Inductive proofs using the well-foundedness of ScPrec are fiddly. The following induction principle simplifies the proofs.

$\boldsymbol{syn\_induction\_thm} =$
$\vdash \forall\ p$
$\bullet\ (\forall\ x\ y\bullet\ p\ (MkAf\ (x,\ y)))$
$\wedge\ (\forall\ vars\ fs\bullet\ (\forall f\bullet\ f \in_g fs \Rightarrow f \in Syntax)$
$\wedge\ (\forall f\bullet\ f \in_g fs \Rightarrow p\ f) \Rightarrow p\ (MkCf\ (vars,\ fs)))$
$\Rightarrow\ (\forall\ x\bullet\ x \in Syntax \Rightarrow p\ x)$

Using this induction principle an induction tactic is defined as follows:

SML

```
fun infos_induction_tac t (a,c) = (
      let val l1 = mk_app (mk_λ (t,c), t)
          and l2 = mk_app (mk_app (mk_const ("∈", ⌜:GS → GS SET → BOOL⌝), t),
                           mk_const ("Syntax", ⌜:GS SET⌝))
      in  let val l3 = mk_∀ (t, mk_⇒ (l2, l1))
      in  LEMMA_T l1 (rewrite_thm_tac o rewrite_rule[])
      THEN DROP_ASM_T l2 ante_tac
      THEN LEMMA_T l3 (rewrite_thm_tac o rewrite_rule[])
      THEN bc_tac [syn_induction_thm]
      THEN rewrite_tac[]
      THEN strip_tac
      end end) (a,c);
```

This tactic expects an argument t of type $TERM$ which is a free variable of type GS whose sole occurrence in the assumptions is in an assumption $\ulcorner_{\mathrm{ML}}\llcorner t \urcorner \in \mathrm{Syntax}\urcorner$, and results in two subgoals, one requiring a proof for atomic and the other for compound formulae (with the benefit of the induction hypothesis in the assumptions).

2.5 Recursion Theorem

The following recursion theorem supports definition by primitive recursion of functions over the syntax. This version does require (or allow) stipulation of a value for elements which are not part of the syntax (which is sometimes necessary). The function defined is total over the type GS, but its value is only known for members of *Syntax*.

```
sc_recursion_lemma =
  ⊢ ∀ af cf
    • ∃ f
      • (∀ m s• f (MkAf (m, s)) = af m s)
          ∧ (∀ vars forms
          • f (MkCf (vars, forms)) = cf (FunImage_g f forms) vars forms)
```

This gets plugged into proof context *'infos* for use in consistency proofs.

SML

```
add_∃_cd_thms [sc_recursion_lemma] "'infos";
set_merge_pcs ["misc21", "'infos"];
```

Definitions of the form supported by the above recursion lemma do not allow sufficient control over the value of a function when applied outside the range of the syntactic constructors. The following version (if it works) is suitable for cases where this level of control is necessary.

$$
\begin{array}{l}
\textbf{\textit{sc\_recursion\_lemma2}} = \\
\quad \vdash \forall\ af\ cf\ ov \\
\quad\quad \bullet \exists\ f \\
\quad\quad\quad \bullet\ (\forall\ m\ s \bullet\ f\ (MkAf\ (m,\ s)) = af\ m\ s) \\
\quad\quad\quad\quad \wedge\ (\forall\ vars\ forms \\
\quad\quad\quad\quad \bullet\ f\ (MkCf\ (vars,\ forms)) = cf\ (FunImage_g\ f\ forms)\ vars\ forms) \\
\quad\quad\quad\quad \wedge\ (\forall\ x \bullet\ \neg\ IsAf\ x \wedge \neg\ IsCf\ x \Rightarrow f\ x = ov\ x)
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{MkTrash}} : GS \rightarrow GS \\
\hline
\forall x \bullet\ MkTrash\ x = if\ IsAf\ x \vee IsCf\ x\ then\ \varnothing_g\ else\ x
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{sc\_recursion\_lemma3}} = \\
\quad \vdash \forall\ af\ cf\ ov \\
\quad\quad \bullet \exists\ f \\
\quad\quad\quad \bullet\ (\forall\ m\ s \bullet\ f\ (MkAf\ (m,\ s)) = af\ m\ s) \\
\quad\quad\quad\quad \wedge\ (\forall\ vars\ forms \\
\quad\quad\quad\quad \bullet\ f\ (MkCf\ (vars,\ forms)) = cf\ (FunImage_g\ f\ forms)\ vars\ forms) \\
\quad\quad\quad\quad \wedge\ (\forall\ x \bullet\ f\ (MkTrash\ x) = ov\ (MkTrash\ x))
\end{array}
$$

This also gets plugged into proof context *'infos* for use in consistency proofs.

SML

```
add_∃_cd_thms [sc_recursion_lemma3] "'infos";
set_merge_pcs ["misc21", "'infos"];
```

2.6 Auxiliary Concepts

Its useful to be able to talk about the free variables in a formula so the definition is given here.

The definition is by recursion over the structure of the syntax.

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{FreeVars}} : GS \rightarrow GS\ SET \\
\hline
(\forall x\ y \bullet \\
\quad\quad FreeVars\ (MkAf\ (x,\ y)) = \{x;\ y\}) \\
\wedge\ (\forall vars\ forms \bullet \\
\quad\quad FreeVars\ (MkCf\ (vars,\ forms)) = \bigcup\ (FunImage_g\ FreeVars\ forms) \setminus (X_g\ vars))
\end{array}
$$

The name *SetReps* is defined as the set of formulae with exactly one free variable which is the empty set. These are the candidate representatives for sets, and represent the set coextensive with the property expressed by the formula. To know what set that is you need to know the domain of discourse (which in the cases of interest here will always be a subset of *SetReps*) and the semantics of formulae, which is defined below.

HOL Constant

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

$$SetReps = \{x \mid x \in Syntax \wedge \exists y \bullet\ FreeVars\ x = \{y\}\}$$

$$\textbf{\textit{setreps\_}}\subseteq\textbf{\textit{\_syntax\_lemma}} =$$
$$\vdash SetReps \subseteq Syntax$$

$$\textbf{\textit{setreps\_}}\subseteq\textbf{\textit{\_syntax\_lemma2}} =$$
$$\vdash V \subseteq SetReps \Rightarrow V \subseteq Syntax$$

3 SEMANTICS

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

3.1 Type Abbreviations

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

The following type abbreviations are introduced:

RV Relation Value - this is the type for the meaning of a formula with free variables. The parameters are the type of the domain of discourse and the type of truth values.

ST Structure = a structure is a domain of discourse (a set) together with a binary relation (the membership relation) over that domain. The membership relation need not (and will not) be boolean. The parameters are the type of the domain of discourse and the type of truth values.

SML

```
declare_infix(300, "∈ᵥ");
declare_type_abbrev("RV", ["'a","'b"], ⌜:(GS, 'a)IX → 'b⌝);
declare_type_abbrev("ST", ["'a","'b"], ⌜:'a SET × ('a, 'b)BR⌝);
```

3.2 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 (which are all membership claims) are obtained from a 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 membership relation for those arguments in the structure. Note that this specification is generic in the type of truth values.

HOL Constant

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

$$\forall \$\leq_t\ (at{:}GS)\ (st{:}('a, 't)\ ST)\ (va{:}(GS, 'a)IX)\bullet\ EvalAf\ \$\leq_t\ at\ st\ va =$$
$$let\ d = Fst\ st$$
$$and\ rv = Snd\ st$$
$$and\ set = AfSet\ at$$
$$and\ mem = AfMem\ at$$
$$in\quad if\ mem \in IxDom\ va \wedge set \in IxDom\ va$$
$$then\ rv\ (IxVal\ va\ mem)\ (IxVal\ va\ set)$$
$$else\ Lub\ \$\leq_t\ \{\}$$

$$\boldsymbol{EvalAf\_MkAf\_lemma} =$$
$$\vdash \forall \leq_t\ mem\ set\ st\ va$$
$$\bullet\ EvalAf\ \leq_t\ (MkAf\ (set, mem))\ st\ va$$
$$= (if\ mem \in IxDom\ va \wedge set \in IxDom\ va$$
$$then\ Snd\ st\ (IxVal\ va\ mem)\ (IxVal\ va\ set)$$
$$else\ Lub\ \leq_t\ \{\})$$

To evaluate a compound formula you must first evaluate the constituent formulae in every context obtainable by modification of those variables which are 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 the available truth values.

The definition of the truth function depends upon the type of truth values, and is therefore a parameter of the semantics.

The relevant definitions for three and four valued truth types are given here.

SML

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

This function is the core of the semantics of the logic, and captures the truth functional character of first order logic, including the quantifiers even in the infinitary case. It should be remembered that though I am working here with four truth values, the logic we are formalising is a classical two valued first order logic. The extra truth values are included to facilitate the discovery of interpretations by making the semantics monotonic and taking least and/or greatest fixed points.

To make this clear I am defining the semantics as a two valued function first so that you can see the intended semantics without the clutter of the extra truth values. The semantics is for a single compound formula constructor which takes a set of formulae and a set of variables and yields the negation of the (infinitary) conjunction of all the instances of the formulae for every possible valuation of the bound variables. This function expects the results of evaluating these formulae as a set of truth values, and returns the truth value of the compound formula.

HOL Constant

$$\boldsymbol{EvalCf\_bool} : BOOL\ SET \to BOOL$$

$$\forall results\bullet\ EvalCf\_bool\ results = F \in results$$

That looks unbelievably simple doesn't it.

When we generalise this to operate with four truth values, you should think of the truth values as sets of boolean truth values ordered by inclusion, i.e. with the empty set as "bottom" and the set of both truth values as "top". Then think of the required evaluation function as arising by mapping the boolean evaluator (*EvalCf_bool*) over the set of choice sets formed from the set of sets of booleans.

HOL Constant

$$\boldsymbol{LiftEvalCf\_bool} : BOOL\ SET\ SET \to BOOL\ SET$$

$$\forall results \bullet\ LiftEvalCf\_bool\ results =$$
$$\{v{:}BOOL \mid \exists w\ f \bullet\ w = FunImage\ f\ results \wedge v = EvalCf\_bool\ w$$
$$\wedge\ \forall x \bullet\ x \in results \wedge (\exists y \bullet\ y \in x) \Rightarrow (f\ x) \in x\}$$

To get the required evaluator we need to modify this to work with type FTV instead of BOOL SET[1].

The conversions are:

HOL Constant

$$\boldsymbol{BoolSet2FTV} : BOOL\ SET \to FTV$$

$$\forall bs \bullet\ BoolSet2FTV\ bs =$$
$$if\ T \in bs$$
$$then\ if\ F \in bs\ then\ fT\ else\ fTrue$$
$$else\ if\ F \in bs\ then\ fFalse\ else\ fB$$

HOL Constant

$$\boldsymbol{FTV2BoolSet} : FTV \to BOOL\ SET$$

$$\forall ftv \bullet\ FTV2BoolSet\ ftv =$$
$$\{x \mid (x = T \wedge fTrue \leq_{t4} ftv) \vee (x = F \wedge fFalse \leq_{t4} ftv\ )\}$$

HOL Constant

$$\boldsymbol{EvalCf2\_ftv} : FTV\ CFE$$

$$\forall results \bullet\ EvalCf2\_ftv\ results =$$
$$BoolSet2FTV\ (LiftEvalCf\_bool\ (FunImage\ FTV2BoolSet\ results))$$

Deriving the result by that means looked like it would be a bit complicated so here's my guess what the result should be:

HOL Constant

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

$$\forall results \bullet\ EvalCf\_ftv\ results =\ Lub\ \$\leq_{t4}\ \{t \mid$$
$$(fFalse \in results \vee fT \in\ results) \wedge t = fTrue$$
$$\vee$$
$$(\neg\ fFalse \in results \wedge \neg\ fB \in results) \wedge t = fFalse$$
$$\}$$

[1] I ask myself "Why isn't FTV just BOOL SET?"

$$
\begin{array}{l}
\textbf{\textit{evalcf\_ftv\_lemma}} = \\
\quad \vdash \forall\ s \\
\quad\quad \bullet\ \mathit{EvalCf\_ftv}\ s \\
\quad\quad\quad = (\mathit{if}\ \mathit{fFalse} \in s \vee \mathit{fT} \in s \\
\quad\quad\quad\quad \mathit{then\ if}\ \neg\ \mathit{fFalse} \in s \wedge \neg\ \mathit{fB} \in s\ \mathit{then}\ \mathit{fT}\ \mathit{else}\ \mathit{fTrue} \\
\quad\quad\quad\quad \mathit{else\ if}\ \neg\ \mathit{fFalse} \in s \wedge \neg\ \mathit{fB} \in s \\
\quad\quad\quad\quad \mathit{then}\ \mathit{fFalse} \\
\quad\quad\quad\quad \mathit{else}\ \mathit{fB}) \\
\\
\textbf{\textit{evalcf\_ftv\_ft\_lemma}} = \\
\quad \vdash \forall\ s \bullet\ \mathit{EvalCf\_ftv}\ s = \mathit{fT} \Leftrightarrow \neg\ \mathit{fFalse} \in s \wedge \mathit{fT} \in s \wedge \neg\ \mathit{fB} \in s \\
\\
\textbf{\textit{evalcf\_ftv\_ft\_lemma1}} = \\
\quad \vdash \forall\ s \bullet\ \mathit{EvalCf\_ftv}\ s = \mathit{fT} \Rightarrow \neg\ \mathit{fFalse} \in s \wedge \mathit{fT} \in s \wedge \neg\ \mathit{fB} \in s \\
\\
\textbf{\textit{evalcf\_ftv\_fb\_lemma}} = \\
\quad \vdash \forall\ s \bullet\ \mathit{EvalCf\_ftv}\ s = \mathit{fB} \Leftrightarrow \neg\ \mathit{fFalse} \in s \wedge \neg\ \mathit{fT} \in s \wedge \mathit{fB} \in s \\
\\
\textbf{\textit{evalcf\_ftv\_fb\_lemma1}} = \\
\quad \vdash \forall\ s \bullet\ \mathit{EvalCf\_ftv}\ s = \mathit{fB} \Rightarrow \neg\ \mathit{fFalse} \in s \wedge \neg\ \mathit{fT} \in s \wedge \mathit{fB} \in s
\end{array}
$$

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

$$
\begin{array}{l}
\textbf{\textit{EvalCf}} : 't\ \mathit{CFE} \to \mathit{GS} \to ('a, 't)\ \mathit{ST} \to ('a, 't)\ \mathit{RV}\ \mathit{SET} \to ('a, 't)\ \mathit{RV} \\
\hline
\forall \mathit{etf}\ f \bullet\ \mathit{EvalCf}\ \mathit{etf}\ f = \lambda \mathit{st}\ \mathit{rvs}\ \mathit{va} \bullet \\
\quad\quad \mathit{let}\ \nu = \mathit{CfVars}\ f \\
\quad\quad \mathit{and}\ V = \mathit{Fst}\ \mathit{st} \\
\quad\quad \mathit{in}\ \mathit{etf}\ \{\mathit{pb} \mid \exists \mathit{rv}\ v \bullet \\
\quad\quad\quad\quad\quad \mathit{rv} \in \mathit{rvs} \\
\quad\quad\quad\quad \wedge\ \mathit{IxDom}\ v = X_g\ \nu \\
\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ v \subseteq V \\
\quad\quad\quad\quad \wedge\ \mathit{pb} = \mathit{rv}\ (\mathit{IxOverRide}\ \mathit{va}\ v)\}
\end{array}
$$

$$
\begin{array}{l}
\mathit{EvalCf\_MkCf\_lemma} = \\
\quad \vdash \forall\ \mathit{etf}\ \mathit{vars}\ \mathit{forms}\ \mathit{st}\ \mathit{rvs}\ \mathit{va} \\
\quad\quad \bullet\ \mathit{EvalCf}\ \mathit{etf}\ (\mathit{MkCf}\ (\mathit{vars}, \mathit{forms}))\ \mathit{st}\ \mathit{rvs}\ \mathit{va} \\
\quad\quad\quad = \mathit{etf} \\
\quad\quad\quad\quad \{\mathit{pb} \\
\quad\quad\quad\quad\quad \mid \exists\ \mathit{rv}\ v \\
\quad\quad\quad\quad\quad\quad \bullet\ \mathit{rv} \in \mathit{rvs} \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IxDom}\ v = X_g\ \mathit{vars} \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ v \subseteq \mathit{Fst}\ \mathit{st} \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{pb} = \mathit{rv}\ (\mathit{IxOverRide}\ \mathit{va}\ v)\}
\end{array}
$$

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

HOL Constant

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

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

The semantics of formulae is then given by:

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.

$$\boldsymbol{evalformfunct\_respect\_thm} =$$
$$\vdash \forall\ cfe\ \leq_t\ st\bullet\ EvalFormFunct\ (cfe, \leq_t, st)\ respects\ ScPrec$$

$$\boldsymbol{evalformfunct\_fixp\_lemma} =$$
$$\vdash \forall\ cfe\ \leq_t\ st$$
$$\bullet\ EvalForm\ (cfe, \leq_t, st)$$
$$= EvalFormFunct\ (cfe, \leq_t, st)\ (EvalForm\ (cfe, \leq_t, st))$$

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

$$
\begin{array}{l}
\textbf{\textit{evalformfunct\_thm2}} = \\
\quad \vdash \forall\ cfe\ \leq_t\ st\ f \\
\quad\quad \bullet\ EvalForm\ (cfe,\ \leq_t,\ st)\ f \\
\quad\quad\quad = (if\ f \in Syntax \\
\quad\quad\quad\quad then \\
\quad\quad\quad\quad\quad if\ IsAf\ f \\
\quad\quad\quad\quad\quad then\ EvalAf\ \leq_t\ f\ st \\
\quad\quad\quad\quad\quad else \\
\quad\quad\quad\quad\quad\quad (let\ rvs \\
\quad\quad\quad\quad\quad\quad\quad\quad = FunImage\ (EvalForm\ (cfe,\ \leq_t,\ st))\ (X_g\ (CfForms\ f)) \\
\quad\quad\quad\quad\quad\quad in\ EvalCf\ cfe\ f\ st\ rvs) \\
\quad\quad\quad\quad else\ \epsilon\ f\bullet\ T)
\end{array}
$$

A similar effect can be obtained more concisely as follows:

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{EvalForm2}} : 't\ CFE \times 't\ REL \times ('a, 't)\ ST \to GS \to ('a, 't)\ RV \\
\hline \\
(\forall cfe\ \$\leq_t\ V\ rv\ mem\ set\ va\bullet\ EvalForm2\ (cfe,\ \$\leq_t,\ (V,\ rv))\ (MkAf\ (mem,\ set))\ va = \\
\quad\quad if\ mem \in IxDom\ va \wedge set \in IxDom\ va \\
\quad\quad then\ rv\ (IxVal\ va\ mem)\ (IxVal\ va\ set) \\
\quad\quad else\ Lub\ \$\leq_t\ \{\}) \\
\ \wedge \\
(\forall cfe\ \$\leq_t\ V\ rv\ \nu\ forms\ va\bullet\ EvalForm2\ (cfe,\ \$\leq_t,\ (V,\ rv))\ (MkCf\ (\nu,\ forms))\ va = \\
\quad\quad cfe\ \{pb \mid \exists rv2\ v\bullet\ rv2 \in FunImage_g\ (EvalForm2\ (cfe,\ \$\leq_t,\ (V,\ rv)))\ forms \\
\quad\quad\quad\quad \wedge\ IxDom\ v = X_g\ \nu \\
\quad\quad\quad\quad \wedge\ IxRan\ v \subseteq V \\
\quad\quad\quad\quad \wedge\ pb = rv2\ (IxOverRide\ va\ v)\}) \\
\ \wedge \\
(\forall cfe\ \$\leq_t\ V\ rv\ x\ va\bullet\ EvalForm2\ (cfe,\ \$\leq_t,\ (V,\ rv))\ (MkTrash\ x)\ va = Lub\ \$\leq_t\ \{\})
\end{array}
$$

However the effect is not identical, and this does not stipulate what happens to elements in the domain type of the function (taking this as GS and treating previous arguments as parameters) but not in the set *Syntax*. This might complicate the problem of establishing the monotonicity of the semantics.

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_MkAf\_lemma}} = \\
\quad \vdash \forall\ cfe\ \leq_t\ st\ set\ mem \\
\quad\quad \bullet\ EvalForm\ (cfe,\ \leq_t,\ st)\ (MkAf\ (set,\ mem)) \\
\quad\quad\quad = EvalAf\ \leq_t\ (MkAf\ (set,\ mem))\ st
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_fT\_lemma}} = \\
\quad \vdash \forall\ V\ y \\
\quad\quad \bullet\ y \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ va \\
\quad\quad\quad \bullet\ \mathit{FreeVars}\ y \subseteq \mathit{IxDom}\ va \\
\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ va \subseteq V \cup \{\varnothing_g\} \\
\quad\quad\quad\quad \wedge\ \mathit{EvalForm}\ (\mathit{EvalCf\_ftv}, \$\leq_{t4},\ V \cup \{\varnothing_g\}, \$\in_v)\ y\ va = fT \\
\quad\quad\quad\quad \Rightarrow (\exists\ x\ y \bullet\ x \in V \cup \{\varnothing_g\} \wedge y \in V \cup \{\varnothing_g\} \wedge x \in_v y = fT)) \\
\\
\textbf{\textit{EvalForm2\_fT\_lemma}} = \\
\quad \vdash \forall\ V\ y \\
\quad\quad \bullet\ y \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ va \\
\quad\quad\quad \bullet\ \mathit{FreeVars}\ y \subseteq \mathit{IxDom}\ va \\
\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ va \subseteq V \cup \{\varnothing_g\} \\
\quad\quad\quad\quad \wedge\ \mathit{EvalForm2}\ (\mathit{EvalCf\_ftv}, \$\leq_{t4},\ V \cup \{\varnothing_g\}, \$\in_v)\ y\ va = fT \\
\quad\quad\quad\quad \Rightarrow (\exists\ x\ y \bullet\ x \in V \cup \{\varnothing_g\} \wedge y \in V \cup \{\varnothing_g\} \wedge x \in_v y = fT))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_fT\_lemma2}} = \\
\quad \vdash \forall\ y \\
\quad\quad \bullet\ y \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ va \\
\quad\quad\quad \bullet\ \mathit{FreeVars}\ y \subseteq \mathit{IxDom}\ va \\
\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ va \subseteq V \cup \{\varnothing_g\} \\
\quad\quad\quad\quad \wedge\ \mathit{EvalForm}\ (\mathit{EvalCf\_ftv}, \$\leq_{t4},\ V, \$\in_v)\ y\ va = fT \\
\quad\quad\quad\quad \Rightarrow (\exists\ x\ y \bullet\ x \in V \cup \{\varnothing_g\} \wedge y \in V \cup \{\varnothing_g\} \wedge x \in_v y = fT)) \\
\\
\textbf{\textit{EvalForm2\_fT\_lemma2}} = \\
\quad \vdash \forall\ V\ y \\
\quad\quad \bullet\ y \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ va \\
\quad\quad\quad \bullet\ \mathit{FreeVars}\ y \subseteq \mathit{IxDom}\ va \\
\quad\quad\quad\quad \wedge\ \mathit{IxRan}\ va \subseteq V \cup \{\varnothing_g\} \\
\quad\quad\quad\quad \wedge\ \mathit{EvalForm2}\ (\mathit{EvalCf\_ftv}, \$\leq_{t4},\ V, \$\in_v)\ y\ va = fT \\
\quad\quad\quad\quad \Rightarrow (\exists\ x\ y \bullet\ x \in V \cup \{\varnothing_g\} \wedge y \in V \cup \{\varnothing_g\} \wedge x \in_v y = fT))
\end{array}
$$

4 MONOTONICITY

4.1 Equivalence of Membership Relations

It proves useful in later proofs to have a notion of equivalence over partial membership relations and to obtain here some lemmas about aspects of the semantics which depend upon a prior notion of membership and give the same result for equivalent relations.

HOL Constant

$$\textbf{\textit{PmrEq}} : 'a\ SET \to ('a, 'b)\ BR \to ('a, 'b)\ BR \to BOOL$$

$$\forall V \bullet\ PmrEq\ V = \lambda r1\ r2 \bullet \forall x\ y \bullet\ x \in V \wedge y \in V \Rightarrow r1\ x\ y = r2\ x\ y$$

$$\textbf{\textit{PmrEq\_EvalForm\_lemma}} =$$
$$\vdash \forall\ cfe\ \leq_t\ V\ W\ f\ r1\ r2$$
$$\bullet\ V \subseteq W \wedge PmrEq\ W\ r1\ r2$$
$$\Rightarrow f \in Syntax$$
$$\Rightarrow (\forall\ z$$
$$\bullet\ IxRan\ z \subseteq W$$
$$\Rightarrow EvalForm\ (cfe, \leq_t, V, r1)\ f\ z$$
$$= EvalForm\ (cfe, \leq_t, V, r2)\ f\ z)$$

$$\textbf{\textit{PmrEq\_EvalForm2\_lemma}} =$$
$$\vdash \forall\ cfe\ \leq_t\ V\ W\ f\ r1\ r2$$
$$\bullet\ V \subseteq W \wedge PmrEq\ W\ r1\ r2$$
$$\Rightarrow f \in Syntax$$
$$\Rightarrow (\forall\ z$$
$$\bullet\ IxRan\ z \subseteq W$$
$$\Rightarrow EvalForm2\ (cfe, \leq_t, V, r1)\ f\ z$$
$$= EvalForm2\ (cfe, \leq_t, V, r2)\ f\ z)$$

4.2 Some Orderings

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

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

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

$$\textbf{\textit{rvo\_lubs\_exist\_thm}} =$$
$$\vdash \forall\ r \bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvO\ r)$$

$$\textbf{\textit{rvo\_glbs\_exist\_thm}} =$$
$$\vdash \forall\ r \bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvO\ r)$$

HOL Constant

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

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

$$\textbf{\textit{rviso\_lubs\_exist\_thm}} =$$
$$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow LubsExist\ (RvIsO\ r)$$

$$\textbf{\textit{rviso\_glbs\_exist\_thm}} =$$
$$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (RvIsO\ r)$$

HOL Constant

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

$$\forall r\bullet\ StO\ r = DerivedOrder\ Snd\ (Pw\ (Pw\ r))$$

$$\textbf{\textit{sto\_lubs\_exist\_thm}} =$$
$$\vdash \forall\ r\bullet\ LubsExist\ r \Rightarrow LubsExist\ (StO\ r)$$

$$\textbf{\textit{sto\_glbs\_exist\_thm}} =$$
$$\vdash \forall\ r\bullet\ GlbsExist\ r \Rightarrow GlbsExist\ (StO\ r)$$

We will be looking for fixed points of the semantics and one approach to this is to take greatest of least fixed points over various subdomains of the formulae of this set theory. This is why we are working with non-standard truth value types, so that we can arrange for the semantics to be monotonic relative to orderings derived from that on the truth values.

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 (though in fact the proof has been completed). 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 predicates.

SML

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

HOL Constant

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

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

$$\mid \textbf{\textit{ccrpou\_}}\leq_{\textbf{\textit{ft3}}}\textbf{\textit{\_thm}} =$$
$$\mid \quad \vdash \mathit{CcRpoU}\ \$\leq_{ft3}$$

Lets now get on with the four valued case.

SML

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

HOL Constant

$$\mid \$\leq_{\textbf{\textit{ft4}}} : ('a \to \mathit{FTV}) \to ('a \to \mathit{FTV}) \to \mathit{BOOL}$$
$$\mid \quad \text{---}$$
$$\mid \$\leq_{ft4} = \mathit{Pw}\ \$\leq_{t4}$$

$$\mid \leq_{\textbf{\textit{ft4}}}\textbf{\textit{\_crpou\_thm}} =$$
$$\mid \quad \vdash \ \mathit{CRpoU}\ \$\leq_{ft4}$$

We need an ordering over variable assignments relative to membership relations which corresponds to degrees of well-definedness of the sets assigned to variables under the membership relation.

It might be helpful to do this separately for the extension and the essence of the sets, so that's what I propose to do here.

We begin with pre-orderings over *SetReps* corresponding to extension.

HOL Constant

$$\mid \textbf{\textit{ExtSRO}} : \mathit{GS}\ \mathit{SET} \times (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to \mathit{GS} \to \mathit{GS} \to \mathit{BOOL}$$
$$\mid \quad \text{---}$$
$$\mid \forall V\ r\bullet\ \mathit{ExtSRO}\ (V,\ r) = \lambda x\ y\bullet \forall z\bullet\ z \in V \Rightarrow r\ z\ x \leq_{t4} r\ z\ y$$

and to essences.

HOL Constant

$$\mid \textbf{\textit{EssSRO}} : \mathit{GS}\ \mathit{SET} \times (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to \mathit{GS} \to \mathit{GS} \to \mathit{BOOL}$$
$$\mid \quad \text{---}$$
$$\mid \forall V\ r\bullet\ \mathit{EssSRO}\ (V,\ r) = \lambda x\ y\bullet \forall z\bullet\ z \in V \Rightarrow r\ x\ z \leq_{t4} r\ y\ z$$

Though I suspect I may only need:

HOL Constant

$$\mid \textbf{\textit{ExsSRO}} : \mathit{GS}\ \mathit{SET} \times (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to \mathit{GS} \to \mathit{GS} \to \mathit{BOOL}$$
$$\mid \quad \text{---}$$
$$\mid \forall s\bullet\ \mathit{ExsSRO}\ s = \mathit{ConjOrder}\ (\mathit{ExtSRO}\ s)\ (\mathit{EssSRO}\ s)$$

From this we obtain a pre-ordering over variable assignments:

HOL Constant

$$\mid \textbf{\textit{ExsVaO}} : ('a,\ \mathit{GS})\ \mathit{IX}\ \mathit{SET} \times (\mathit{GS}\ \mathit{SET} \times (\mathit{GS},\ \mathit{FTV})\mathit{BR})$$
$$\mid \quad \to ('a,\ \mathit{GS})\ \mathit{IX} \to ('a,\ \mathit{GS})\ \mathit{IX} \to \mathit{BOOL}$$
$$\mid \quad \text{---}$$
$$\mid \forall d\ s\bullet\ \mathit{ExsVaO}\ (d,\ s) = \mathit{IxO2}\ (d,\ (\mathit{ExsSRO}\ s))$$

The following function is given to determine appropriate sets of indexed sets for use with this ordering. That is, for any domain V the set of V-valued indexed sets (there is no constraint on the domain for any set can be a variable and we use the indexed sets primarily for variable assignments).

HOL Constant

$$\boldsymbol{V2IxSet} : GS\ SET \to ('a,\ GS)\ IX\ SET$$

$$\forall V \bullet\ V2IxSet\ V = \{ix \mid IxRan\ ix \subseteq V\}$$

$$\boldsymbol{exsvao\_ixoverride\_lemma} =$$
$$\vdash \forall\ V\ \$\in_v\ x\ y\ v$$
$$\bullet\ ExsVaO\ (V2IxSet\ V,\ V,\ \$\in_v)\ x\ y \wedge IxRan\ v \subseteq V$$
$$\Rightarrow ExsVaO\ (V2IxSet\ V,\ V,\ \$\in_v)\ (IxOverRide\ x\ v)\ (IxOverRide\ y\ v)$$

$$\boldsymbol{exsvao\_ixoverride\_lemma2} =$$
$$\vdash \forall\ V\ W\ \$\in_v\ x\ y\ v$$
$$\bullet\ ExsVaO\ (V2IxSet\ V,\ W,\ \$\in_v)\ x\ y \wedge IxRan\ v \subseteq V$$
$$\Rightarrow ExsVaO\ (V2IxSet\ V,\ W,\ \$\in_v)\ (IxOverRide\ x\ v)\ (IxOverRide\ y\ v)$$

4.3 Monotonicity Results

First we prove the monotonicity of *EvalForm*.

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

$$\boldsymbol{evalaf\_increasing\_lemma} =$$
$$\vdash \forall\ tr\ g \bullet\ CRpoU\ tr \Rightarrow Increasing\ (StO\ tr)\ (RvO\ tr)\ (EvalAf\ tr\ g)$$

To get a monotonicity 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 mapping for each fixed domain of discourse and each particular formula, which take a membership structure (an interpretation of set theory over the gived domain) and returns the relation represented by the formula 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

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

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

HOL Constant

$$\boldsymbol{MonoEvalForm2} : 't\ CFE \times 't\ REL \times 'a\ SET \times GS \to ('a, 't)\ BR \to ('a, 't)\ RV$$

$$\forall c\ r\ s\ g\ ris \bullet\ MonoEvalForm2\ (c,\ r,\ s,\ g)\ ris = EvalForm2\ (c,\ r,\ (s,\ ris))\ g$$

$$
\begin{array}{l}
\textbf{\textit{monoevalform\_increasing\_lemma}} = \\
\quad \vdash \forall\ c\ r\ s\ g \\
\quad\quad \bullet\ \mathit{CRpoU}\ r \wedge \mathit{Increasing}\ (\mathit{SetO}\ r)\ r\ c \\
\quad\quad\quad \Rightarrow \mathit{Increasing}\ (\mathit{Pw}\ (\mathit{Pw}\ r))\ (\mathit{RvO}\ r)\ (\mathit{MonoEvalForm}\ (c,\ r,\ s,\ g))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{evalform\_increasing\_thm}} = \\
\quad \vdash \forall\ c\ r\ s\ g \\
\quad\quad \bullet\ \mathit{CRpoU}\ r \wedge \mathit{Increasing}\ (\mathit{SetO}\ r)\ r\ c \\
\quad\quad\quad \Rightarrow \mathit{Increasing}\ (\mathit{Pw}\ (\mathit{Pw}\ r))\ (\mathit{RvO}\ r)\ (\lambda\ \mathit{ris} \bullet\ \mathit{EvalForm}\ (c,\ r,\ s,\ \mathit{ris})\ g)
\end{array}
$$

We will also need to prove monotonicity of formula evaluation relative to the extension and essences of the variable assignment providing the context for the evaluation.

$$
\begin{array}{l}
\textbf{\textit{evalform\_increasing\_thm2}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ g \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge g \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow \mathit{Increasing} \\
\quad\quad\quad\quad (\mathit{ExsVaO}\ (\mathit{V2IxSet}\ V,\ V,\ \$\in_v)) \\
\quad\quad\quad\quad \$\leq_{t4} \\
\quad\quad\quad\quad (\mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leq_{t4},\ V,\ \$\in_v)\ g)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{evalform\_increasing\_thm3}} = \\
\quad \vdash \forall\ V\ \$\in_v\ g \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge g \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow \mathit{Increasing} \\
\quad\quad\quad\quad (\mathit{ExsVaO}\ (\mathit{V2IxSet}\ (V \cup \{\varnothing_g\}),\ V \cup \{\varnothing_g\},\ \$\in_v)) \\
\quad\quad\quad\quad \$\leq_{t4} \\
\quad\quad\quad\quad (\mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leq_{t4},\ V,\ \$\in_v)\ g)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{evalform\_increasing\_thm4}} = \\
\quad \vdash \forall\ V\ W\ \$\in_v\ g \\
\quad\quad \bullet\ V \subseteq W \wedge V \subseteq \mathit{Syntax} \wedge g \in \mathit{Syntax} \\
\quad\quad\quad \Rightarrow \mathit{Increasing} \\
\quad\quad\quad\quad (\mathit{ExsVaO}\ (\mathit{V2IxSet}\ W,\ W,\ \$\in_v)) \\
\quad\quad\quad\quad \$\leq_{t4} \\
\quad\quad\quad\quad (\mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leq_{t4},\ V,\ \$\in_v)\ g)
\end{array}
$$

5 EXTENSIONALITY

5.1 Extension and Essence, Compatibility and OverDefinedness

The terminology is a bit uncertain here. These lemmas about the semantics are proven in order to obtain extensionality results for fixed points of the semantic functor.

HOL Constant

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

$$Extension = CombC$$

We now define a notion of 'same extension' over some domain.

Two sets have the same extension over some domain if they have the same members in that domain.

HOL Constant

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

$$\forall V\ r\bullet\ SameExt\ V\ r = \lambda x\ y\bullet\ \forall z\bullet\ z \in V \Rightarrow r\ z\ x = r\ z\ y$$

HOL Constant

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

$$Essence = CombI$$

Two sets have the same essence over some domain if they are members of the same sets in that domain.

HOL Constant

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

$$\forall V\ r\bullet\ SameEss\ V\ r = \lambda x\ y\bullet\ \ \forall z\bullet\ z \in V \Rightarrow r\ x\ z = r\ y\ z$$

$$\boldsymbol{sameext\_equiv\_thm} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ SameExt\ V\ r)$$

$$\boldsymbol{sameess\_equiv\_thm} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ SameEss\ V\ r)$$

$$\boldsymbol{sameext\_refl\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x\bullet\ x \in V \Rightarrow SameExt\ V\ r\ x\ x$$

$$\boldsymbol{sameess\_refl\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x\bullet\ x \in V \Rightarrow SameEss\ V\ r\ x\ x$$

$$\boldsymbol{sameext\_sym\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (SameExt\ V\ r\ x\ y \Leftrightarrow SameExt\ V\ r\ y\ x)$$

$$\boldsymbol{sameess\_sym\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (SameEss\ V\ r\ x\ y \Leftrightarrow SameEss\ V\ r\ y\ x)$$

$$
\begin{array}{l}
\textbf{\textit{evalform\_ext\_lemma}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ V \subseteq SetReps \\
\quad\quad\quad \Rightarrow (\forall\ z \\
\quad\quad\quad \bullet\ z \in Syntax \\
\quad\quad\quad\quad \Rightarrow (\forall\ x\ y \\
\quad\quad\quad\quad \bullet\ IxDom\ x = IxDom\ y \\
\quad\quad\quad\quad\quad \wedge\ FreeVars\ z \subseteq IxDom\ x \\
\quad\quad\quad\quad\quad \wedge\ IxRan\ x \subseteq V \\
\quad\quad\quad\quad\quad \wedge\ IxRan\ y \subseteq V \\
\quad\quad\quad\quad\quad \wedge\ (\forall\ v \\
\quad\quad\quad\quad\quad \bullet\ v \in IxDom\ x \\
\quad\quad\quad\quad\quad\quad \Rightarrow SameExt\ V\ r\ (IxVal\ x\ v)\ (IxVal\ y\ v) \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ SameEss\ V\ r\ (IxVal\ x\ v)\ (IxVal\ y\ v)) \\
\quad\quad\quad\quad \Rightarrow EvalForm\ (EvalCf\_ftv, \$\leq_{t4},\ V,\ r)\ z\ x \\
\quad\quad\quad\quad\quad = EvalForm\ (EvalCf\_ftv, \$\leq_{t4},\ V,\ r)\ z\ y))
\end{array}
$$

Unfortunaately that last lemma is too weak for the purposes for which is was intended. To obtain a stronger lemma it is necessary to have concepts weaker than *SameExt* and *SameEss*, as follows:

To obtain a stronger lemma we define the notion of 'compatible extension'.

Two sets have compatible extensions over some domain if they do not definitely disagree about what their members are, where a disagreement is definite if at some point either is overdefined or if neither is undefined. This is defined in terms of compatibility of truth value sets, which is defined in [1].

HOL Constant

$$\textbf{\textit{CompExt}} : GS\ SET \rightarrow (GS,\ FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL$$

$$\forall V\ r \bullet\ CompExt\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ z\ x;\ r\ z\ y\} \in CompFTV$$

HOL Constant

$$\textbf{\textit{CoCompExt}} : GS\ SET \rightarrow (GS,\ FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL$$

$$\forall V\ r \bullet\ CoCompExt\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ z\ x;\ r\ z\ y\} \in CoCompFTV$$

Two sets have compatible essences over some domain if they do not definitely disagree about which sets they are members of.

HOL Constant

$$\textbf{\textit{CompEss}} : GS\ SET \rightarrow (GS,\ FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL$$

$$\forall V\ r \bullet\ CompEss\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CompFTV$$

HOL Constant

$$\textbf{\textit{CoCompEss}} : GS\ SET \rightarrow (GS,\ FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL$$

$$\forall V\ r \bullet\ CoCompEss\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CoCompFTV$$

We also need to be able to talk about relations which are nowhere overdefined. In fact we need to be able to talk about parts of relations, e.g. particular extensions and essences in these terms.

The following is a property, either of an extension or an essence, which tells you whether it anwhere takes the value fT. The property is parameterised by the domain of discourse.

HOL Constant

$$\boldsymbol{OverDefinedL} : GS\ SET \to (GS \to FTV) \to BOOL$$

$$\forall V\ xe \bullet\ OverDefinedL\ V\ xe \Leftrightarrow \exists y \bullet\ y \in V \wedge xe\ y = fT$$

$$\boldsymbol{OverDefinedL\_\leq_{t4}\_lemma} =$$
$$\vdash \forall\ V\ xe1\ xe2$$
$$\bullet \neg\ OverDefinedL\ V\ xe1 \wedge PwS\ V\ \$\leq_{t4}\ xe2\ xe1 \Rightarrow \neg\ OverDefinedL\ V\ xe2$$

The following property of relations could have been defined in terms of the above, but actually it was defined first and is simpler left as it is.

HOL Constant

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

$$\forall V\ r \bullet\ OverDefined\ V\ r \Leftrightarrow \exists x\ y \bullet\ x \in V \wedge y \in V \wedge r\ x\ y = fT$$

HOL Constant

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

$$\forall V\ W\ r \bullet\ OverDefinedEss\ V\ W\ r \Leftrightarrow \exists x \bullet\ x \in V \wedge OverDefinedL\ W\ (Essence\ r\ x)$$

HOL Constant

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

$$\forall V\ W\ r \bullet\ OverDefinedExt\ V\ W\ r \Leftrightarrow \exists x \bullet\ x \in V \wedge OverDefinedL\ W\ (Extension\ r\ x)$$

and the duals:

HOL Constant

$$\boldsymbol{UnderDefinedL} : GS\ SET \to (GS \to FTV) \to BOOL$$

$$\forall V\ xe \bullet\ UnderDefinedL\ V\ xe \Leftrightarrow \exists y \bullet\ y \in V \wedge xe\ y = fB$$

HOL Constant

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

$$\forall V\ r \bullet\ UnderDefined\ V\ r \Leftrightarrow \exists x\ y \bullet\ x \in V \wedge y \in V \wedge r\ x\ y = fB$$

$$\textbf{\textit{compext\_refl\_lemma}} =$$
$$\vdash \forall\ V\ r\ x\bullet \neg\ \mathit{OverDefined}\ V\ r \Rightarrow x \in V \Rightarrow \mathit{CompExt}\ V\ r\ x\ x$$

$$\textbf{\textit{cocompext\_refl\_lemma}} =$$
$$\vdash \forall\ V\ r\ x\bullet \neg\ \mathit{UnderDefined}\ V\ r \Rightarrow x \in V \Rightarrow \mathit{CoCompExt}\ V\ r\ x\ x$$

$$\textbf{\textit{compess\_refl\_lemma}} =$$
$$\vdash \forall\ V\ r\ x\bullet \neg\ \mathit{OverDefined}\ V\ r \Rightarrow x \in V \Rightarrow \mathit{CompEss}\ V\ r\ x\ x$$

$$\textbf{\textit{cocompess\_refl\_lemma}} =$$
$$\vdash \forall\ V\ r\ x\bullet \neg\ \mathit{UnderDefined}\ V\ r \Rightarrow x \in V \Rightarrow \mathit{CoCompEss}\ V\ r\ x\ x$$

$$\textbf{\textit{compext\_sym\_lemma}} =$$
$$\vdash \forall\ V\ r$$
$$\bullet \neg\ \mathit{OverDefined}\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (\mathit{CompExt}\ V\ r\ x\ y \Leftrightarrow \mathit{CompExt}\ V\ r\ y\ x))$$

$$\textbf{\textit{cocompext\_sym\_lemma}} =$$
$$\vdash \forall\ V\ r$$
$$\bullet \neg\ \mathit{UnderDefined}\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (\mathit{CoCompExt}\ V\ r\ x\ y \Leftrightarrow \mathit{CoCompExt}\ V\ r\ y\ x))$$

$$\textbf{\textit{compess\_sym\_lemma}} =$$
$$\vdash \forall\ V\ r$$
$$\bullet \neg\ \mathit{OverDefined}\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (\mathit{CompEss}\ V\ r\ x\ y \Leftrightarrow \mathit{CompEss}\ V\ r\ y\ x))$$

$$\textbf{\textit{cocompess\_sym\_lemma}} =$$
$$\vdash \forall\ V\ r$$
$$\bullet \neg\ \mathit{UnderDefined}\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (\mathit{CoCompEss}\ V\ r\ x\ y \Leftrightarrow \mathit{CoCompEss}\ V\ r\ y\ x))$$

The following condition expresses the possibility that two set representatives might end up representing the same set after further iterations of the semantic functor in approaching a least fixed point.

This intended for use in proving that a total least fixed point will always be extensional, through a lemma which states that, under certain conditions, elements which are compatible will still have the same extension after one further application of the semantic functor.

To be compatible two elements must have the same essences and extensions, but this does not suffice. It is also necessary to state that there is no disagreement about whether the element to which they might be refined by iteration of the semantic functor is a member of itself.

HOL Constant

$$\textbf{\textit{Compatible}} : \mathit{GS}\ \mathit{SET} \rightarrow (\mathit{GS}, \mathit{FTV})\mathit{BR} \rightarrow \mathit{GS} \rightarrow \mathit{GS} \rightarrow \mathit{BOOL}$$

$$\forall V\ r\bullet\ \mathit{Compatible}\ V\ r = \lambda x\ y\bullet \{r\ x\ y;\ r\ y\ x;\ r\ x\ x;\ r\ y\ y\} \in \mathit{CompFTV}$$
$$\wedge\ \mathit{CompEss}\ V\ r\ x\ y \wedge \mathit{CompExt}\ V\ r\ x\ y$$

```
Compatible_lemma1 =
  ⊢ ∀ V r x y• Compatible V r x y ⇔ {r x y; r y x; r x x; r y y} ∈ CompFTV
          ∧ (∀ z• z ∈ V ⇒
                {r z x; r z y} ∈ CompFTV
              ∧ {r x z; r y z} ∈ CompFTV)
```

6 PROOF CONTEXTS

SML

```
(* add_pc_thms "'infos" [evalcf_ftv_ft_lemma, evalcf_ftv_fb_lemma]; *)

(* add_pc_thms "'infos" [get_spec ⌜Extension⌝, get_spec ⌜Essence⌝]; *)
commit_pc "'infos";

force_new_pc "infos";
merge_pcs ["misc2", "'infos"] "infos";
commit_pc "infos";

force_new_pc "infos1";
merge_pcs ["misc21", "'infos"] "infos1";
commit_pc "infos1";
```

SML

```
set_flag ("subgoal_package_quiet", false);
```

7 The Theory infos

7.1 Parents

misc2

7.2 Constants

$\boldsymbol{MkAf}$	$GS \times GS \to GS$
$\boldsymbol{IsAf}$	$GS \to BOOL$
$\boldsymbol{AfSet}$	$GS \to GS$
$\boldsymbol{AfMem}$	$GS \to GS$
$\boldsymbol{MkCf}$	$GS \times GS \to GS$
$\boldsymbol{IsCf}$	$GS \to BOOL$
$\boldsymbol{CfVars}$	$GS \to GS$
$\boldsymbol{CfForms}$	$GS \to GS$
$\boldsymbol{MkNot}$	$GS \to GS$
$\boldsymbol{RepClosed}$	$GS\ \mathbb{P} \to BOOL$
$\boldsymbol{Syntax}$	$GS\ \mathbb{P}$
$\boldsymbol{RepOpen}$	$GS\ \mathbb{P} \to BOOL$
$\boldsymbol{CoSyntax}$	$GS\ \mathbb{P}$
$\boldsymbol{ScPrec}$	$(GS,\ BOOL)\ BR$
$\boldsymbol{ScPrec2}$	$(GS,\ BOOL)\ BR$
$\boldsymbol{MkTrash}$	$GS \to GS$
$\boldsymbol{FreeVars}$	$GS \to GS\ \mathbb{P}$
$\boldsymbol{SetReps}$	$GS\ \mathbb{P}$
$\boldsymbol{EvalAf}$	$('t,\ BOOL)\ BR \to GS \to ('a,\ 't)\ ST \to ('a,\ 't)\ RV$
$\boldsymbol{EvalCf\_bool}$	$BOOL\ CFE$
$\boldsymbol{LiftEvalCf\_bool}$	$BOOL\ \mathbb{P}\ CFE$
$\boldsymbol{BoolSet2FTV}$	$BOOL\ \mathbb{P} \to FTV$
$\boldsymbol{FTV2BoolSet}$	$FTV \to BOOL\ \mathbb{P}$
$\boldsymbol{EvalCf2\_ftv}$	$FTV\ CFE$
$\boldsymbol{EvalCf\_ftv}$	$FTV\ CFE$
$\boldsymbol{EvalCf}$	$'t\ CFE \to GS \to ('a,\ 't)\ ST \to ('a,\ 't)\ RV\ CFE$
$\boldsymbol{EvalFormFunct}$	$'t\ CFE \times ('t,\ BOOL)\ BR \times ('a,\ 't)\ ST \to (GS \to ('a,\ 't)\ RV) \to GS \to ('a,\ 't)\ RV$
$\boldsymbol{EvalForm}$	$'t\ CFE \times ('t,\ BOOL)\ BR \times ('a,\ 't)\ ST \to GS \to ('a,\ 't)\ RV$
$\boldsymbol{EvalForm2}$	$'t\ CFE \times ('t,\ BOOL)\ BR \times ('a,\ 't)\ ST \to GS \to ('a,\ 't)\ RV$
$\boldsymbol{PmrEq}$	$'a\ \mathbb{P} \to (('a,\ 'b)\ BR,\ BOOL)\ BR$
$\boldsymbol{RvO}$	$('b,\ BOOL)\ BR \to (('a,\ 'b)\ RV,\ BOOL)\ BR$
$\boldsymbol{RvIsO}$	$('b,\ BOOL)\ BR \to (('a,\ 'b)\ RV\ IS,\ BOOL)\ BR$
$\boldsymbol{StO}$	$('b,\ BOOL)\ BR \to (('a,\ 'b)\ ST,\ BOOL)\ BR$
$\$\leq_{\boldsymbol{ft3}}$	$(('a,\ BOOL)\ IX,\ BOOL)\ BR$
$\$\leq_{\boldsymbol{ft4}}$	$('a \to FTV,\ BOOL)\ BR$
$\boldsymbol{ExtSRO}$	$(GS,\ FTV)\ ST \to (GS,\ BOOL)\ BR$
$\boldsymbol{EssSRO}$	$(GS,\ FTV)\ ST \to (GS,\ BOOL)\ BR$
$\boldsymbol{ExsSRO}$	$(GS,\ FTV)\ ST \to (GS,\ BOOL)\ BR$

ExsVaO $('a,\ GS)\ IX\ \mathbb{P} \times (GS,\ FTV)\ ST \to (('a,\ GS)\ IX,\ BOOL)\ BR$
V2IxSet $GS\ \mathbb{P} \to ('a,\ GS)\ IX\ \mathbb{P}$
MonoEvalForm $'t\ CFE \times ('t,\ BOOL)\ BR \times 'a\ \mathbb{P} \times GS \to ('a, 't)\ BR \to ('a, 't)\ RV$
MonoEvalForm2 $'t\ CFE \times ('t,\ BOOL)\ BR \times 'a\ \mathbb{P} \times GS \to ('a, 't)\ BR \to ('a, 't)\ RV$
Extension $(GS,\ FTV)\ BR \to (GS,\ FTV)\ BR$
SameExt $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
Essence $(GS,\ FTV)\ BR \to (GS,\ FTV)\ BR$
SameEss $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
CompExt $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
CoCompExt $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
CompEss $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
CoCompEss $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$
OverDefinedL $GS\ \mathbb{P} \to BOOL\ IS \to BOOL$
OverDefined $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to BOOL$
OverDefinedEss $(GS\ \mathbb{P},\ (GS,\ FTV)\ BR \to BOOL)\ BR$
OverDefinedExt $(GS\ \mathbb{P},\ (GS,\ FTV)\ BR \to BOOL)\ BR$
UnderDefinedL $GS\ \mathbb{P} \to BOOL\ IS \to BOOL$
UnderDefined $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to BOOL$
Compatible $GS\ \mathbb{P} \to (GS,\ FTV)\ BR \to (GS,\ BOOL)\ BR$

7.3 Type Abbreviations

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

7.4 Fixity

Right Infix 300:
$\in_{\boldsymbol{v}} \quad \leq_{\boldsymbol{ft3}} \leq_{\boldsymbol{ft4}}$

7.5 Definitions

MkAf $\vdash \forall\ lr\bullet\ MkAf\ lr = Nat_g\ 0 \mapsto_g Fst\ lr \mapsto_g Snd\ lr$
IsAf $\vdash \forall\ t\bullet\ IsAf\ t \Leftrightarrow (\exists\ lr\bullet\ t = MkAf\ lr)$
AfSet $\vdash AfSet = (\lambda\ x\bullet\ snd\ (snd\ x))$
AfMem $\vdash AfMem = (\lambda\ x\bullet\ fst\ (snd\ x))$
MkCf $\vdash \forall\ vc\bullet\ MkCf\ vc = Nat_g\ 1 \mapsto_g Fst\ vc \mapsto_g Snd\ vc$
IsCf $\vdash \forall\ t\bullet\ IsCf\ t \Leftrightarrow (\exists\ vc\bullet\ t = MkCf\ vc)$
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 $\vdash \forall\ s$
$\bullet\ RepClosed\ s$
$\Leftrightarrow (\forall\ s2\ m\bullet\ MkAf\ (s2,\ m) \in s)$
$\wedge (\forall\ vars\ fs\bullet\ X_g\ fs \subseteq s \Rightarrow MkCf\ (vars,\ fs) \in s)$

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

RepOpen $\vdash \forall\ s$
$\bullet\ RepOpen\ s$
$\Leftrightarrow (\forall\ vars\ fs\bullet\ MkCf\ (vars,\ fs) \in s \Rightarrow X_g\ fs \subseteq s)$

CoSyntax $\vdash CoSyntax = \bigcup \{x | RepOpen\ x\}$

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

ScPrec2 $\vdash \forall\ \alpha\ \gamma$
$\bullet\ ScPrec2\ \alpha\ \gamma$
$\Leftrightarrow (\exists\ ord\ fs\bullet\ \alpha \in_g fs \wedge \gamma = MkCf\ (ord,\ fs))$

MkTrash $\vdash \forall\ x\bullet\ MkTrash\ x = (if\ IsAf\ x \vee IsCf\ x\ then\ \varnothing_g\ else\ x)$

FreeVars $\vdash (\forall\ x\ y\bullet\ FreeVars\ (MkAf\ (x,\ y)) = \{x;\ y\})$
$\wedge (\forall\ vars\ forms$
$\bullet\ FreeVars\ (MkCf\ (vars,\ forms))$
$= \bigcup (FunImage_g\ FreeVars\ forms) \setminus X_g\ vars)$

SetReps $\vdash SetReps = \{x | x \in Syntax \wedge (\exists\ y\bullet\ FreeVars\ x = \{y\})\}$

EvalAf $\vdash \forall \leq_t\ at\ st\ va$
$\bullet\ EvalAf \leq_t at\ st\ va$
$= (let\ d = Fst\ st$
$and\ rv = Snd\ st$
$and\ set = AfSet\ at$
$and\ mem = AfMem\ at$
$in\ if\ mem \in IxDom\ va \wedge set \in IxDom\ va$
$then\ rv\ (IxVal\ va\ mem)\ (IxVal\ va\ set)$
$else\ Lub \leq_t \{\})$

EvalCf_bool $\vdash \forall\ results\bullet\ EvalCf\_bool\ results \Leftrightarrow F \in results$

LiftEvalCf_bool $\vdash \forall\ results$
$\bullet\ LiftEvalCf\_bool\ results$
$= \{v$
$|\exists\ w\ f$
$\bullet\ w = FunImage\ f\ results$
$\wedge (v \Leftrightarrow EvalCf\_bool\ w)$
$\wedge (\forall\ x$
$\bullet\ x \in results \wedge (\exists\ y\bullet\ y \in x) \Rightarrow f\ x \in x)\}$

BoolSet2FTV $\vdash \forall\ bs$
$\bullet\ BoolSet2FTV\ bs$
$= (if\ T \in bs$
$then\ if\ F \in bs\ then\ fT\ else\ fTrue$
$else\ if\ F \in bs$
$then\ fFalse$
$else\ fB)$

FTV2BoolSet $\vdash \forall\ ftv$

$$
\begin{array}{l}
\quad \bullet\ FTV2BoolSet\ ftv \\
\quad\quad = \{x \\
\quad\quad |(x \Leftrightarrow T) \wedge fTrue \leq_{t4} ftv \\
\quad\quad\quad \vee (x \Leftrightarrow F) \wedge fFalse \leq_{t4} ftv\}
\end{array}
$$

EvalCf2_ftv

$$
\begin{array}{l}
\vdash \forall\ results \\
\quad \bullet\ EvalCf2\_ftv\ results \\
\quad\quad = BoolSet2FTV \\
\quad\quad\ (LiftEvalCf\_bool \\
\quad\quad\quad (FunImage\ FTV2BoolSet\ results))
\end{array}
$$

EvalCf_ftv

$$
\begin{array}{l}
\vdash \forall\ results \\
\quad \bullet\ EvalCf\_ftv\ results \\
\quad\quad = Lub \\
\quad\quad\ \$\leq_{t4} \\
\quad\quad\ \{t \\
\quad\quad\quad |(fFalse \in results \vee fT \in results) \\
\quad\quad\quad\quad \wedge\ t = fTrue \\
\quad\quad\quad\quad \vee (\neg\ fFalse \in results \wedge \neg\ fB \in results) \\
\quad\quad\quad\quad \wedge\ t = fFalse\}
\end{array}
$$

EvalCf

$$
\begin{array}{l}
\vdash \forall\ etf\ f \\
\quad \bullet\ EvalCf\ \ etf\ f \\
\quad\quad = (\lambda\ st\ rvs\ va \\
\quad\quad \bullet\ (let\ \nu = CfVars\ f\ \ and\ \ V = Fst\ st \\
\quad\quad\quad in\ \ etf \\
\quad\quad\quad\ \{pb \\
\quad\quad\quad\quad |\exists\ rv\ v \\
\quad\quad\quad\quad\quad \bullet\ rv \in rvs \\
\quad\quad\quad\quad\quad\quad \wedge\ IxDom\ v = X_g\ \nu \\
\quad\quad\quad\quad\quad\quad \wedge\ IxRan\ v \subseteq V \\
\quad\quad\quad\quad\quad\quad \wedge\ pb = rv\ (IxOverRide\ va\ v)\}))
\end{array}
$$

EvalFormFunct

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

EvalForm

$$
\begin{array}{l}
\vdash \forall\ cfe \leq_t st \\
\quad \bullet\ EvalForm\ (cfe, \leq_t, st) \\
\quad\quad = fix\ (EvalFormFunct\ (cfe, \leq_t, st))
\end{array}
$$

EvalForm2

$$
\begin{array}{l}
\vdash (\forall\ cfe \leq_t V\ rv\ mem\ set\ va \\
\quad \bullet\ EvalForm2\ (cfe, \leq_t, V, rv)\ (MkAf\ (mem, set))\ va \\
\quad\quad = (if\ mem \in IxDom\ va \wedge set \in IxDom\ va \\
\quad\quad\ then\ rv\ (IxVal\ va\ mem)\ (IxVal\ va\ set) \\
\quad\quad\ else\ Lub \leq_t \{\})) \\
\quad \wedge (\forall\ cfe \leq_t V\ rv\ \nu\ forms\ va \\
\quad \bullet\ EvalForm2\ (cfe, \leq_t, V, rv)\ (MkCf\ (\nu, forms))\ va
\end{array}
$$

$$
\begin{array}{ll}
 & = cfe \\
 & \quad \{pb \\
 & \quad\quad | \exists\ rv2\ v \\
 & \quad\quad\quad \bullet\ rv2 \\
 & \quad\quad\quad\quad \in FunImage_g \\
 & \quad\quad\quad\quad\quad (EvalForm2\ (cfe, \leq_t, V, rv)) \\
 & \quad\quad\quad\quad\quad forms \\
 & \quad\quad\quad\quad \wedge\ IxDom\ v = X_g\ \nu \\
 & \quad\quad\quad\quad \wedge\ IxRan\ v \subseteq V \\
 & \quad\quad\quad\quad \wedge\ pb = rv2\ (IxOverRide\ va\ v)\}) \\
 & \quad \wedge (\forall\ cfe \leq_t V\ rv\ x\ va \\
 & \quad \bullet\ EvalForm2\ (cfe, \leq_t, V, rv)\ (MkTrash\ x)\ va \\
 & \quad\quad = Lub\ \leq_t \{\}) \\
\textbf{\textit{PmrEq}} & \vdash \forall\ V \\
 & \quad \bullet\ PmrEq\ V \\
 & \quad\quad = (\lambda\ r1\ r2 \\
 & \quad\quad \bullet \forall\ x\ y \bullet\ x \in V \wedge y \in V \Rightarrow r1\ x\ y = r2\ x\ y) \\
\textbf{\textit{RvO}} & \vdash \forall\ r \bullet\ RvO\ r = Pw\ r \\
\textbf{\textit{RvIsO}} & \vdash \forall\ r \bullet\ RvIsO\ r = IsEO\ (RvO\ r) \\
\textbf{\textit{StO}} & \vdash \forall\ r \bullet\ StO\ r = DerivedOrder\ Snd\ (Pw\ (Pw\ r)) \\
\textbf{\textit{}}\leq_{\textbf{\textit{ft3}}} & \vdash \$\leq_{ft3} = Pw\ \$\leq_{t3} \\
\textbf{\textit{}}\leq_{\textbf{\textit{ft4}}} & \vdash \$\leq_{ft4} = Pw\ \$\leq_{t4} \\
\textbf{\textit{ExtSRO}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ ExtSRO\ (V, r) \\
 & \quad\quad = (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow r\ z\ x \leq_{t4} r\ z\ y) \\
\textbf{\textit{EssSRO}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ EssSRO\ (V, r) \\
 & \quad\quad = (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow r\ x\ z \leq_{t4} r\ y\ z) \\
\textbf{\textit{ExsSRO}} & \vdash \forall\ s \bullet\ ExsSRO\ s = ConjOrder\ (ExtSRO\ s)\ (EssSRO\ s) \\
\textbf{\textit{ExsVaO}} & \vdash \forall\ d\ s \bullet\ ExsVaO\ (d, s) = IxO2\ (d, ExsSRO\ s) \\
\textbf{\textit{V2IxSet}} & \vdash \forall\ V \bullet\ V2IxSet\ V = \{ix | IxRan\ ix \subseteq V\} \\
\textbf{\textit{MonoEvalForm}} & \vdash \forall\ c\ r\ s\ g\ ris \\
 & \quad \bullet\ MonoEvalForm\ (c, r, s, g)\ ris \\
 & \quad\quad = EvalForm\ (c, r, s, ris)\ g \\
\textbf{\textit{MonoEvalForm2}} & \\
 & \vdash \forall\ c\ r\ s\ g\ ris \\
 & \quad \bullet\ MonoEvalForm2\ (c, r, s, g)\ ris \\
 & \quad\quad = EvalForm2\ (c, r, s, ris)\ g \\
\textbf{\textit{Extension}} & \vdash Extension = CombC \\
\textbf{\textit{SameExt}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ SameExt\ V\ r = (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow r\ z\ x = r\ z\ y) \\
\textbf{\textit{Essence}} & \vdash Essence = CombI \\
\textbf{\textit{SameEss}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ SameEss\ V\ r = (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow r\ x\ z = r\ y\ z) \\
\textbf{\textit{CompExt}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ CompExt\ V\ r \\
 & \quad\quad = (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow \{r\ z\ x; r\ z\ y\} \in CompFTV) \\
\textbf{\textit{CoCompExt}} & \vdash \forall\ V\ r \\
 & \quad \bullet\ CoCompExt\ V\ r \\
 & \quad\quad = (\lambda\ x\ y \\
 & \quad\quad \bullet \forall\ z \bullet\ z \in V \Rightarrow \{r\ z\ x; r\ z\ y\} \in CoCompFTV)
\end{array}
$$

CompEss $\vdash \forall\ V\ r$
$\bullet\ CompEss\ V\ r$
$= (\lambda\ x\ y \bullet \forall\ z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CompFTV)$

CoCompEss $\vdash \forall\ V\ r$
$\bullet\ CoCompEss\ V\ r$
$= (\lambda\ x\ y$
$\bullet \forall\ z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CoCompFTV)$

OverDefinedL $\vdash \forall\ V\ xe \bullet\ OverDefinedL\ V\ xe \Leftrightarrow (\exists\ y \bullet\ y \in V \wedge xe\ y = fT)$

OverDefined $\vdash \forall\ V\ r$
$\bullet\ OverDefined\ V\ r$
$\Leftrightarrow (\exists\ x\ y \bullet\ x \in V \wedge y \in V \wedge r\ x\ y = fT)$

OverDefinedEss
$\vdash \forall\ V\ W\ r$
$\bullet\ OverDefinedEss\ V\ W\ r$
$\Leftrightarrow (\exists\ x \bullet\ x \in V \wedge OverDefinedL\ W\ (Essence\ r\ x))$

OverDefinedExt
$\vdash \forall\ V\ W\ r$
$\bullet\ OverDefinedExt\ V\ W\ r$
$\Leftrightarrow (\exists\ x \bullet\ x \in V \wedge OverDefinedL\ W\ (Extension\ r\ x))$

UnderDefinedL
$\vdash \forall\ V\ xe \bullet\ UnderDefinedL\ V\ xe \Leftrightarrow (\exists\ y \bullet\ y \in V \wedge xe\ y = fB)$

UnderDefined $\vdash \forall\ V\ r$
$\bullet\ UnderDefined\ V\ r$
$\Leftrightarrow (\exists\ x\ y \bullet\ x \in V \wedge y \in V \wedge r\ x\ y = fB)$

Compatible $\vdash \forall\ V\ r$
$\bullet\ Compatible\ V\ r$
$= (\lambda\ x\ y$
$\bullet\ \{r\ x\ y;\ r\ y\ x;\ r\ x\ x;\ r\ y\ y\} \in CompFTV$
$\wedge\ CompEss\ V\ r\ x\ y$
$\wedge\ CompExt\ V\ r\ x\ y)$

7.6 Theorems

Is_clauses $\vdash (\forall\ x \bullet\ IsAf\ (MkAf\ x))$
$\wedge (\forall\ x \bullet \neg\ IsAf\ (MkCf\ x))$
$\wedge (\forall\ x \bullet \neg\ IsCf\ (MkAf\ x))$
$\wedge (\forall\ x \bullet\ IsCf\ (MkCf\ x))$

syn_proj_clauses
$\vdash (\forall\ s\ m \bullet\ AfMem\ (MkAf\ (m,\ s)) = m)$
$\wedge (\forall\ s\ m \bullet\ AfSet\ (MkAf\ (m,\ s)) = s)$
$\wedge (\forall\ v\ f \bullet\ CfVars\ (MkCf\ (v,\ f)) = v)$
$\wedge (\forall\ v\ f \bullet\ CfForms\ (MkCf\ (v,\ f)) = f)$

syn_con_inv_fc_clauses
$\vdash \forall\ p$
$\bullet\ (IsAf\ p \Rightarrow MkAf\ (AfMem\ p,\ AfSet\ p) = p)$
$\wedge\ (IsCf\ p \Rightarrow MkCf\ (CfVars\ p,\ CfForms\ p) = p)$

syn_con_eq_clauses
$\vdash \forall\ p1\ p2$
$\bullet\ (MkAf\ p1 = MkAf\ p2 \Leftrightarrow p1 = p2)$
$\wedge\ (MkCf\ p1 = MkCf\ p2 \Leftrightarrow p1 = p2)$

syn_con_neq_clauses

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

is_fc_clauses1

$\vdash \forall\ x$
$\bullet\ (IsAf\ x \Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m)))$
$\wedge\ (IsCf\ x \Rightarrow (\exists\ vars\ fs\bullet\ x = MkCf\ (vars,\ fs)))$

Is_not_cases $\vdash \forall\ x\bullet \neg\ IsAf\ x \vee \neg\ IsCf\ x$

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

$\vdash (\forall\ s2\ m\bullet\ MkAf\ (s2,\ m) \in Syntax)$
$\wedge\ (\forall\ vars\ fs$
$\bullet\ (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in Syntax)$
$\Rightarrow\ MkCf\ (vars,\ fs) \in Syntax)$

repopen_cosyntax_thm

$\vdash \forall\ vars\ fs$
$\bullet\ MkCf\ (vars,\ fs) \in CoSyntax$
$\Rightarrow (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in CoSyntax)$

repclosed_syntax_thm2

$\vdash (\forall\ s2\ m\bullet\ MkAf\ (s2,\ m) \in Syntax)$
$\wedge\ (\forall\ vars\ fs$
$\bullet\ (\forall\ x\bullet\ x \in_g fs \Rightarrow x \in Syntax)$
$\Rightarrow\ MkCf\ (vars,\ fs) \in Syntax)$

repopen_cosyntax_thm2

$\vdash \forall\ vars\ fs$
$\bullet\ MkCf\ (vars,\ fs) \in CoSyntax$
$\Rightarrow (\forall\ x\bullet\ x \in_g fs \Rightarrow x \in CoSyntax)$

repclosed_syntax_lemma1

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

repopen_cosyntax_lemma1

$\vdash \forall\ s\bullet\ RepOpen\ s \Rightarrow s \subseteq CoSyntax$

repclosed_syntax_lemma2

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

repopen_cosyntax_lemma2

$\vdash \forall\ p\bullet\ RepOpen\ \{x|p\ x\} \Rightarrow (\forall\ x\bullet\ p\ x \Rightarrow x \in CoSyntax)$

well_founded_ScPrec_thm

$\vdash well\_founded\ ScPrec$

well_founded_ScPrec2_thm

$\vdash well\_founded\ ScPrec2$

well_founded_tcScPrec2_thm

$\vdash well\_founded\ (tc\ ScPrec2)$

syntax_disj_thm

$\vdash \forall\ x$
$\bullet\ x \in Syntax$
$\Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m))$
$\vee\ (\exists\ vars\ fs$
$\bullet\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax)$
$\wedge\ x = MkCf\ (vars,\ fs))$

syntax_cases_thm

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

syntax_cases_fc_clauses

$\vdash \forall\ x$

$\bullet\ x \in Syntax$
$\Rightarrow (\neg\ IsAf\ x \Rightarrow IsCf\ x) \wedge (\neg\ IsCf\ x \Rightarrow IsAf\ x)$

is_fc_clauses

$\vdash \forall\ x$
$\bullet\ x \in Syntax$
$\Rightarrow (IsAf\ x \Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m)))$
$\wedge\ (IsCf\ x$
$\Rightarrow (\exists\ vars\ fs$
$\bullet\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax)$
$\wedge\ x = MkCf\ (vars,\ fs)))$

MkAf_∈_Syntax_lemma

$\vdash \forall\ x\ y\bullet\ MkAf\ (x,\ y) \in Syntax$

¬∅g_∈_syntax_lemma

$\vdash \neg\ \varnothing_g \in Syntax$

¬∅g_∈_syntax_lemma2

$\vdash \forall\ x\bullet\ x \in Syntax \Rightarrow \neg\ x = \varnothing_g$

¬∅g_∈_syntax_lemma3

$\vdash \forall\ V\ x\bullet\ x \in V \wedge V \subseteq Syntax \Rightarrow \neg\ x = \varnothing_g$

is_fc_clauses2

$\vdash \forall\ x$
$\bullet\ x \in Syntax$
$\Rightarrow IsCf\ x$
$\Rightarrow (\forall\ y\bullet\ y \in_g CfForms\ x \Rightarrow y \in Syntax)$

syn_comp_fc_clauses

$\vdash \forall\ v\ f$
$\bullet\ MkCf\ (v,\ f) \in Syntax \Rightarrow (\forall\ y\bullet\ y \in_g f \Rightarrow y \in Syntax)$

scprec_fc_clauses

$\vdash \forall\ \alpha\ \gamma\ vars\ fs$
$\bullet\ \gamma \in Syntax$
$\Rightarrow \gamma = MkCf\ (vars,\ fs) \wedge \alpha \in_g fs$
$\Rightarrow ScPrec\ \alpha\ \gamma$

scprec2_fc_clauses

$\vdash \forall\ \alpha\ \gamma\ vars\ fs$
$\bullet\ \gamma = MkCf\ (vars,\ fs) \wedge \alpha \in_g fs \Rightarrow ScPrec2\ \alpha\ \gamma$

scprec_fc_clauses2

$\vdash \forall\ t$
$\bullet\ t \in Syntax$
$\Rightarrow IsCf\ t$
$\Rightarrow (\forall\ f\bullet\ f \in_g CfForms\ t \Rightarrow ScPrec\ f\ t)$

scprec2_fc_clauses2

$\vdash \forall\ t\bullet\ IsCf\ t \Rightarrow (\forall\ f\bullet\ f \in_g CfForms\ t \Rightarrow ScPrec2\ f\ t)$

syn_induction_thm

$\vdash \forall\ p$
$\bullet\ (\forall\ x\ y\bullet\ p\ (MkAf\ (x,\ y)))$
$\wedge\ (\forall\ vars\ fs$
$\bullet\ (\forall\ f\bullet\ f \in_g fs \Rightarrow f \in Syntax)$
$\wedge\ (\forall\ f\bullet\ f \in_g fs \Rightarrow p\ f)$
$\Rightarrow p\ (MkCf\ (vars,\ fs)))$
$\Rightarrow (\forall\ x\bullet\ x \in Syntax \Rightarrow p\ x)$

sc_recursion_lemma

$\vdash \forall\ af\ cf$

$\bullet\ \exists\ f$
$\bullet\ (\forall\ m\ s\bullet\ f\ (MkAf\ (m,\ s)) = af\ m\ s)$
$\wedge\ (\forall\ vars\ forms$
$\bullet\ f\ (MkCf\ (vars,\ forms))$
$= cf\ (FunImage_g\ f\ forms)\ vars\ forms)$

sc_recursion_lemma3

$\vdash\ \forall\ af\ cf\ ov$
$\bullet\ \exists\ f$
$\bullet\ (\forall\ m\ s\bullet\ f\ (MkAf\ (m,\ s)) = af\ m\ s)$
$\wedge\ (\forall\ vars\ forms$
$\bullet\ f\ (MkCf\ (vars,\ forms))$
$= cf\ (FunImage_g\ f\ forms)\ vars\ forms)$
$\wedge\ (\forall\ x\bullet\ f\ (MkTrash\ x) = ov\ (MkTrash\ x))$

setreps_⊆_syntax_lemma

$\vdash\ SetReps \subseteq Syntax$

setreps_⊆_syntax_lemma2

$\vdash\ V \subseteq SetReps \Rightarrow V \subseteq Syntax$

¬∅g_∈_setreps_lemma

$\vdash\ \neg\ \varnothing_g \in SetReps$

evalcf_ftv_lemma

$\vdash\ \forall\ s$
$\bullet\ EvalCf\_ftv\ s$
$= (if\ fFalse \in s \vee fT \in s$
$then$
$if\ \neg\ fFalse \in s \wedge \neg\ fB \in s\ then\ fT\ else\ fTrue$
$else\ if\ \neg\ fFalse \in s \wedge \neg\ fB \in s$
$then\ fFalse$
$else\ fB)$

evalcf_ftv_ft_lemma

$\vdash\ \forall\ s$
$\bullet\ EvalCf\_ftv\ s = fT$
$\Leftrightarrow\ \neg\ fFalse \in s \wedge fT \in s \wedge \neg\ fB \in s$

evalcf_ftv_ft_lemma1

$\vdash\ \forall\ s$
$\bullet\ EvalCf\_ftv\ s = fT$
$\Rightarrow\ \neg\ fFalse \in s \wedge fT \in s \wedge \neg\ fB \in s$

evalcf_ftv_fb_lemma

$\vdash\ \forall\ s$
$\bullet\ EvalCf\_ftv\ s = fB$
$\Leftrightarrow\ \neg\ fFalse \in s \wedge \neg\ fT \in s \wedge fB \in s$

evalcf_ftv_fb_lemma1

$\vdash\ \forall\ s$
$\bullet\ EvalCf\_ftv\ s = fB$
$\Rightarrow\ \neg\ fFalse \in s \wedge \neg\ fT \in s \wedge fB \in s$

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

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

evalformfunct_thm2

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

EvalForm_fT_lemma

$\vdash\ \forall\ V\ y$
• $y\ \in\ Syntax$
$\Rightarrow\ (\forall\ va$
• $FreeVars\ y\ \subseteq\ IxDom\ va$
$\wedge\ IxRan\ va\ \subseteq\ V\ \cup\ \{\varnothing_g\}$
$\wedge\ EvalForm$
$(EvalCf\_ftv,\ \$\leq_{t4},\ V\ \cup\ \{\varnothing_g\},\ \$\in_v)$
y
va
$= fT$
$\Rightarrow\ (\exists\ x\ y$
• $x\ \in\ V\ \cup\ \{\varnothing_g\}$
$\wedge\ y\ \in\ V\ \cup\ \{\varnothing_g\}$
$\wedge\ y\ \in_v\ x = fT))$

EvalForm_fT_lemma2

$\vdash\ \forall\ y$
• $y\ \in\ Syntax$
$\Rightarrow\ (\forall\ va$
• $FreeVars\ y\ \subseteq\ IxDom\ va$

$\wedge$ *IxRan va* $\subseteq$ $V \cup \{\varnothing_g\}$
$\wedge$ *EvalForm*
(*EvalCf_ftv*, $\$\leq_{t4}$, V, $\$\in_v$)
y
va
$= fT$
$\Rightarrow (\exists\ x\ y$
$\bullet\ x \in V \cup \{\varnothing_g\}$
$\wedge\ y \in V \cup \{\varnothing_g\}$
$\wedge\ y \in_v x = fT))$

EvalForm2_fT_lemma

$\vdash \forall\ V\ y$
$\bullet\ y \in Syntax$
$\Rightarrow (\forall\ va$
$\bullet$ *FreeVars y* $\subseteq$ *IxDom va*
$\wedge$ *IxRan va* $\subseteq$ $V \cup \{\varnothing_g\}$
$\wedge$ *EvalForm2*
(*EvalCf_ftv*, $\$\leq_{t4}$, $V \cup \{\varnothing_g\}$, $\$\in_v$)
y
va
$= fT$
$\Rightarrow (\exists\ x\ y$
$\bullet\ x \in V \cup \{\varnothing_g\}$
$\wedge\ y \in V \cup \{\varnothing_g\}$
$\wedge\ x \in_v y = fT))$

EvalForm2_fT_lemma2

$\vdash \forall\ V\ y$
$\bullet\ y \in Syntax$
$\Rightarrow (\forall\ va$
$\bullet$ *FreeVars y* $\subseteq$ *IxDom va*
$\wedge$ *IxRan va* $\subseteq$ $V \cup \{\varnothing_g\}$
$\wedge$ *EvalForm2*
(*EvalCf_ftv*, $\$\leq_{t4}$, V, $\$\in_v$)
y
va
$= fT$
$\Rightarrow (\exists\ x\ y$
$\bullet\ x \in V \cup \{\varnothing_g\}$
$\wedge\ y \in V \cup \{\varnothing_g\}$
$\wedge\ x \in_v y = fT))$

PmrEq_EvalForm_lemma

$\vdash \forall$ *cfe* $\leq_t$ *V W f r1 r2*
$\bullet\ V \subseteq W \wedge PmrEq\ W\ r1\ r2$
$\Rightarrow f \in Syntax$
$\Rightarrow (\forall\ z$
$\bullet$ *IxRan z* $\subseteq$ *W*
$\Rightarrow$ *EvalForm* (*cfe*, $\leq_t$, V, *r1*) *f z*
$=$ *EvalForm* (*cfe*, $\leq_t$, V, *r2*) *f z*)

PmrEq_EvalForm2_lemma

$\vdash \forall$ *cfe* $\leq_t$ *V W f r1 r2*
$\bullet\ V \subseteq W \wedge PmrEq\ W\ r1\ r2$

$\Rightarrow f \in Syntax$
$\Rightarrow (\forall\ z$
$\bullet\ IxRan\ z \subseteq W$
$\Rightarrow EvalForm2\ (cfe, \leq_t, V, r1)\ f\ z$
$= EvalForm2\ (cfe, \leq_t, V, r2)\ f\ z)$

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

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

ccrpou_$\leq_{ft3}$_thm

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

$\leq_{ft4}$_crpou_thm

$\vdash CRpoU\ \$\leq_{ft4}$

exsvao_ixoverride_lemma

$\vdash \forall\ V\ \$\in_v\ x\ y\ v$
$\bullet\ ExsVaO\ (V2IxSet\ V, V, \$\in_v)\ x\ y \wedge IxRan\ v \subseteq V$
$\Rightarrow ExsVaO$
$(V2IxSet\ V, V, \$\in_v)$
$(IxOverRide\ x\ v)$
$(IxOverRide\ y\ v)$

exsvao_ixoverride_lemma2

$\vdash \forall\ V\ W\ \$\in_v\ x\ y\ v$
$\bullet\ ExsVaO\ (V2IxSet\ V, W, \$\in_v)\ x\ y \wedge IxRan\ v \subseteq V$
$\Rightarrow ExsVaO$
$(V2IxSet\ V, W, \$\in_v)$
$(IxOverRide\ x\ v)$
$(IxOverRide\ y\ v)$

evalcf_ftv_increasing_lemma

$\vdash Increasing\ (SetO\ \$\leq_{t4})\ \$\leq_{t4}\ EvalCf\_ftv$

evalform_increasing_thm2

$\vdash \forall\ (V, \$\in_v)\ g$
$\bullet\ V \subseteq Syntax \wedge g \in Syntax$
$\Rightarrow Increasing$
$(ExsVaO\ (V2IxSet\ V, V, \$\in_v))$
$\$\leq_{t4}$
$(EvalForm\ (EvalCf\_ftv, \$\leq_{t4}, V, \$\in_v)\ g)$

evalform_increasing_thm3

$\vdash \forall\ V\ \$\in_v\ g$
$\bullet\ V \subseteq Syntax \wedge g \in Syntax$
$\Rightarrow Increasing$
$(ExsVaO\ (V2IxSet\ (V \cup \{\varnothing_g\}), V \cup \{\varnothing_g\}, \$\in_v))$
$\$\leq_{t4}$
$(EvalForm\ (EvalCf\_ftv, \$\leq_{t4}, V, \$\in_v)\ g)$

evalform_increasing_thm4

$\vdash \forall\ V\ W\ \$\in_v\ g$
$\bullet\ V \subseteq W \wedge V \subseteq Syntax \wedge g \in Syntax$
$\Rightarrow Increasing$
$(ExsVaO\ (V2IxSet\ W,\ W,\ \$\in_v))$
$\$\leq_{t4}$
$(EvalForm\ (EvalCf\_ftv,\ \$\leq_{t4},\ V,\ \$\in_v)\ g)$

sameext_equiv_thm

$\vdash \forall\ V\ r\bullet\ Equiv\ (V,\ SameExt\ V\ r)$

sameess_equiv_thm

$\vdash \forall\ V\ r\bullet\ Equiv\ (V,\ SameEss\ V\ r)$

sameext_refl_lemma

$\vdash \forall\ V\ r\ x\bullet\ x \in V \Rightarrow SameExt\ V\ r\ x\ x$

sameess_refl_lemma

$\vdash \forall\ V\ r\ x\bullet\ x \in V \Rightarrow SameEss\ V\ r\ x\ x$

sameext_sym_lemma

$\vdash \forall\ V\ r\ x\ y$
$\bullet\ x \in V \wedge y \in V \Rightarrow (SameExt\ V\ r\ x\ y \Leftrightarrow SameExt\ V\ r\ y\ x)$

samess_sym_lemma

$\vdash \forall\ V\ r\ x\ y$
$\bullet\ x \in V \wedge y \in V \Rightarrow (SameEss\ V\ r\ x\ y \Leftrightarrow SameEss\ V\ r\ y\ x)$

OverDefinedL_$\leq_{t4}$_lemma

$\vdash \forall\ V\ xe1\ xe2$
$\bullet \neg\ OverDefinedL\ V\ xe1 \wedge PwS\ V\ \$\leq_{t4}\ xe2\ xe1$
$\Rightarrow \neg\ OverDefinedL\ V\ xe2$

compext_refl_lemma

$\vdash \forall\ V\ r\ x\bullet \neg\ OverDefined\ V\ r \Rightarrow x \in V \Rightarrow CompExt\ V\ r\ x\ x$

cocompext_refl_lemma

$\vdash \forall\ V\ r\ x$
$\bullet \neg\ UnderDefined\ V\ r \Rightarrow x \in V \Rightarrow CoCompExt\ V\ r\ x\ x$

compess_refl_lemma

$\vdash \forall\ V\ r\ x\bullet \neg\ OverDefined\ V\ r \Rightarrow x \in V \Rightarrow CompEss\ V\ r\ x\ x$

cocompess_refl_lemma

$\vdash \forall\ V\ r\ x$
$\bullet \neg\ UnderDefined\ V\ r \Rightarrow x \in V \Rightarrow CoCompEss\ V\ r\ x\ x$

compext_sym_lemma

$\vdash \forall\ V\ r$
$\bullet \neg\ OverDefined\ V\ r$
$\Rightarrow (\forall\ x\ y$
$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (CompExt\ V\ r\ x\ y \Leftrightarrow CompExt\ V\ r\ y\ x))$

cocompext_sym_lemma

$\vdash \forall\ V\ r$
$\bullet \neg\ UnderDefined\ V\ r$
$\Rightarrow (\forall\ x\ y$
$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (CoCompExt\ V\ r\ x\ y \Leftrightarrow CoCompExt\ V\ r\ y\ x))$

compess_sym_lemma

$\vdash \forall\ V\ r$
$\bullet \neg\ OverDefined\ V\ r$
$\Rightarrow (\forall\ x\ y$

$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (CompEss\ V\ r\ x\ y \Leftrightarrow CompEss\ V\ r\ y\ x))$

cocompess_sym_lemma

$\vdash \forall\ V\ r$
$\bullet \neg\ UnderDefined\ V\ r$
$\Rightarrow (\forall\ x\ y$
$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (CoCompEss\ V\ r\ x\ y \Leftrightarrow CoCompEss\ V\ r\ y\ x))$

***Compatible_lemma*1**

$\vdash \forall\ V\ r\ x\ y$
$\bullet\ Compatible\ V\ r\ x\ y$
$\Leftrightarrow \{r\ x\ y;\ r\ y\ x;\ r\ x\ x;\ r\ y\ y\} \in CompFTV$
$\wedge (\forall\ z$
$\bullet\ z \in V$
$\Rightarrow \{r\ z\ x;\ r\ z\ y\} \in CompFTV$
$\wedge \{r\ x\ z;\ r\ y\ z\} \in CompFTV)$

8 INDEX