Infinitarily Definable Sets

Roger Bishop Jones

Abstract

This is my third approach to set theory conceived as a maximal consistent theory of comprehension. It differs from the previous attempt (in t024) by simplification of the treatment of infinitary logic, allowing only a single binary relation.

Created: 2008/07/11

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

http://www.rbjones.com/rbjpub/pp/doc/t026.pdf

Id: t026.doc,v 1.20 2012/08/11 21:01:52 rbj Exp

Contents

To Do

-
-

References

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

[2] 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 infinitary first order set theory (*ifos*). Given any membership structure, i.e. a domain of discourse and a membership relation over that domain, each formula of *ifos* with one free variable will define a subset of the domain of discourse (you may prefer to think of these as classes, since in the domains we work with they will mostly have very large size).

If we take as our domain of discourse the formulae of *ifos*, then the semantics of *ifos* provides a functor which given one membership relation over that domain will yield another membership relation. If this functor had a fixed point then it would provide a rich interpretation of *ifos*. Over the entire set of formulae of *ifos* there can be no such fixed point, for the resulting interpretation would include paradoxical collections such as the Russell set.

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

It is the purpose of this work to see whether obtain models for rich non-well-founded set theories can be obtained in this way.

2 INFINITARY SET THEORY

SML

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

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 are just membership predicates. So its just an ordered pair of 'terms' (and in our case that just means variable names) tagged with a zero.

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

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

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

HOL Constant

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

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

HOL Constant

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

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

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

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

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

HOL Constant

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

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

HOL Constant

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

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

$$\boldsymbol{Is\_clauses} =$$

$$\begin{aligned}
&\vdash (\forall\ x \bullet\ IsAf\ (MkAf\ x)) \\
&\quad \wedge (\forall\ x \bullet \neg\ IsAf\ (MkCf\ x)) \\
&\quad \wedge (\forall\ x \bullet \neg\ IsCf\ (MkAf\ x)) \\
&\quad \wedge (\forall\ x \bullet\ IsCf\ (MkCf\ x))
\end{aligned}$$

$$\boldsymbol{Is\_not\_fc\_clauses} =$$

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

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

$$\boldsymbol{RepClosed}: GS\ SET \to BOOL$$

$$\begin{array}{ll} \forall\ s \bullet\ RepClosed\ s \Leftrightarrow & \\ \quad (\forall\ s2\ m \bullet\ MkAf\ (s2,\ m) \in s) & \\ \wedge \quad (\forall\ vars\ fs \bullet\ X_g\ fs \subseteq s \Rightarrow MkCf\ (vars,\ fs) \in s) & \end{array}$$

The dual concept is also used:

HOL Constant

$$\boldsymbol{RepOpen}: GS\ SET \to BOOL$$

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

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

HOL Constant

$$\boldsymbol{Syntax} : GS\ SET$$

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

The dual probably isn't useful but I've put it in for symmetry.

HOL Constant

$$\boldsymbol{CoSyntax} : GS\ SET$$

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

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

$$\boldsymbol{repopen\_\subseteq\_cosyntax\_thm} =$$
$$\vdash \forall\ s \bullet\ RepOpen\ s \Rightarrow s \subseteq CoSyntax$$

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).
- A recursion theorem which supports definition of primitive recursive functions over the syntax.

repclosed_syntax_lemma =

$$\vdash \mathit{RepClosed}\ \mathit{Syntax}$$

repclosed_syntax_thm =

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

repclosed_syntax_thm2 =

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

repclosed_syntax_lemma1 =

$$\vdash \forall\ s\bullet\ \mathit{RepClosed}\ s \Rightarrow \mathit{Syntax} \subseteq s$$

repclosed_syntax_lemma2 =

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

2.1.3 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 : $\mathit{GS}\ \mathit{REL}$

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

HOL Constant

ScPrec2 : $\mathit{GS}\ \mathit{REL}$

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

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

$$\textbf{\textit{well\_founded\_ScPrec\_thm}} =$$
$$\vdash\ well\_founded\ ScPrec$$

$$\textbf{\textit{well\_founded\_tcScPrec\_thm}} =$$
$$\vdash\ well\_founded\ (tc\ ScPrec)$$

$$\textbf{\textit{ScPrec2\_tc\_}}\in\textbf{\textit{\_thm}} =$$
$$\vdash \forall\ x\ y\bullet\ ScPrec2\ x\ y \Rightarrow tc\ \$\in_g\ x\ y$$

$$\textbf{\textit{well\_founded\_ScPrec2\_thm}} =$$
$$\vdash\ well\_founded\ ScPrec2$$

$$\textbf{\textit{well\_founded\_tcScPrec2\_thm}} =$$
$$\vdash\ well\_founded\ (tc\ ScPrec2)$$

SML

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

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.

$$\textbf{\textit{syntax\_disj\_thm}} =$$
$$\vdash \forall\ x$$
$$\bullet\ x \in Syntax$$
$$\Rightarrow (\exists\ s\ m\bullet\ x = MkAf\ (s,\ m))$$
$$\vee\ (\exists\ vars\ fs\bullet\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs))$$

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

$$\textbf{\textit{syntax\_cases\_fc\_clauses}} =$$
$$\vdash \forall\ x\bullet\ x \in Syntax \Rightarrow (\neg\ IsAf\ x \Rightarrow IsCf\ x) \wedge (\neg\ IsCf\ x \Rightarrow IsAf\ x)$$

$$
\begin{array}{l}
\boldsymbol{is\_fc\_clauses} = \\
\quad \vdash \forall\ x \\
\quad\quad \bullet\ x \in Syntax \\
\quad\quad\quad \Rightarrow (IsAf\ x \Rightarrow (\exists\ s\ m \bullet\ x = MkAf\ (s,\ m))) \\
\quad\quad\quad\quad \wedge\ (IsCf\ x \\
\quad\quad\quad\quad\quad \Rightarrow (\exists\ vars\ fs \\
\quad\quad\quad\quad\quad \bullet\ (\forall\ y \bullet\ y \in_g fs \Rightarrow y \in Syntax) \wedge x = MkCf\ (vars,\ fs)))
\end{array}
$$

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

$$
\begin{array}{l}
\boldsymbol{is\_fc\_clauses2} = \\
\quad \vdash \forall\ x \bullet\ x \in Syntax \Rightarrow IsCf\ x \Rightarrow (\forall\ y \bullet\ y \in_g CfForms\ x \Rightarrow y \in Syntax)
\end{array}
$$

$$
\begin{array}{l}
\neg \varnothing_{\boldsymbol{g}}\boldsymbol{\_\in\_syntax\_lemma} = \\
\quad \vdash \neg\ \varnothing_g \in Syntax
\end{array}
$$

$$
\begin{array}{l}
\neg \varnothing_{\boldsymbol{g}}\boldsymbol{\_\in\_syntax\_lemma2} = \\
\quad \vdash \forall\ x \bullet\ x \in Syntax \Rightarrow \neg\ x = \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\neg \varnothing_{\boldsymbol{g}}\boldsymbol{\_\in\_syntax\_lemma3} = \\
\quad \vdash \forall\ V\ x \bullet\ x \in V \wedge V \subseteq Syntax \Rightarrow \neg\ x = \varnothing_g
\end{array}
$$

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

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

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

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

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

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

```
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 *'ifos* for use in consistency proofs.

SML

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

HOL Constant

```
FreeVars2 : GS → GS SET
```

```
(∀x y•
      FreeVars2 (MkAf (x, y)) = {x; y})
∧ (∀vars forms•
      FreeVars2 (MkCf (vars, forms)) = ⋃ (FunImage_g FreeVars2 forms) \ (X_g vars))
```

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

```
syn_induction_thm =
  ⊢ ∀ p
    • (∀ x y• p (MkAf (x, y)))
      ∧ (∀ vars fs• (∀f• f ∈_g fs ⇒ f ∈ Syntax)
      ∧ (∀f• f ∈_g fs ⇒ p f) ⇒ p (MkCf (vars, fs)))
        ⇒ (∀ x• x ∈ Syntax ⇒ p x)
```

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

SML

```
fun ifos_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 $\ulcorner_{\mathrm{ML}}\ulcorner t \urcorner \in \mathrm{Syntax}\urcorner$, 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).

2.1.4 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. This is acheived by defining a functor of which the required function is a fixed point.

HOL Constant

$$\boldsymbol{FreeVarsFunct} : (GS \rightarrow GS\ SET) \rightarrow (GS \rightarrow GS\ SET)$$

$$\begin{array}{l} FreeVarsFunct = \lambda fv\ f \bullet \\ \quad if\ f \in Syntax \\ \quad then\ if\ IsAf\ f \\ \quad\quad then\ \{AfMem\ f;\ AfSet\ f\} \\ \quad\quad else \bigcup (FunImage_g\ fv\ (CfForms\ f)) \setminus (X_g\ (CfVars\ f)) \\ \quad else\ \epsilon x \bullet T \end{array}$$

HOL Constant

$$\boldsymbol{FreeVars} : (GS \rightarrow GS\ SET)$$

$$FreeVars = fix\ FreeVarsFunct$$

$$\begin{array}{l} \boldsymbol{freevarsfunct\_respect\_thm} = \\ \quad \vdash FreeVarsFunct\ respects\ ScPrec \end{array}$$

$$\begin{array}{l} \boldsymbol{freevarsfunct\_fixp\_lemma} = \\ \quad \vdash FreeVars = FreeVarsFunct\ FreeVars \end{array}$$

$$\begin{array}{l} \boldsymbol{freevarsfunct\_thm} = \\ \quad \vdash FreeVars \\ \quad\quad = (\lambda\ f \\ \quad\quad \bullet\ if\ f \in Syntax \\ \quad\quad\quad then \\ \quad\quad\quad\quad if\ IsAf\ f \\ \quad\quad\quad\quad then\ \{AfMem\ f;\ AfSet\ f\} \\ \quad\quad\quad\quad else \bigcup (FunImage_g\ FreeVars\ (CfForms\ f)) \setminus X_g\ (CfVars\ f) \\ \quad\quad\quad else\ \epsilon\ x \bullet\ T) \end{array}$$

$$\begin{array}{l} \boldsymbol{freevarsfunct\_thm2} = \\ \quad \vdash \forall\ f \\ \quad\quad \bullet\ FreeVars\ f \\ \quad\quad\quad = (if\ f \in Syntax \\ \quad\quad\quad\quad then \\ \quad\quad\quad\quad\quad if\ IsAf\ f \\ \quad\quad\quad\quad\quad then\ \{AfMem\ f;\ AfSet\ f\} \\ \quad\quad\quad\quad\quad else \bigcup (FunImage_g\ FreeVars\ (CfForms\ f)) \setminus X_g\ (CfVars\ f) \\ \quad\quad\quad\quad else\ \epsilon\ x \bullet\ T) \end{array}$$

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

$$\boldsymbol{SetReps} : GS\ SET$$

$$SetReps = \{x \mid x \in Syntax \wedge FreeVars\ x = \{\varnothing_g\}\}$$

setreps_⊆_syntax_lemma =

$$\vdash SetReps \subseteq Syntax$$

setreps_⊆_syntax_lemma2 =

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

2.2 Semantics

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

2.2.1 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⌝);
```

2.2.2 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

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

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

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

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 one is probably now history, but I'm not ready to delete it yet!

HOL Constant

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

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

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 above 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.

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

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

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

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

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

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

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

$$\boldsymbol{evalcf\_ftv\_ft\_lemma} =$$
$$\vdash \forall s \bullet EvalCf\_ftv\ s = fT \Leftrightarrow \neg fFalse \in s \wedge fT \in s \wedge \neg fB \in s$$

$$\boldsymbol{evalcf\_ftv\_ft\_lemma1} =$$
$$\vdash \forall s \bullet EvalCf\_ftv\ s = fT \Rightarrow \neg fFalse \in s \wedge fT \in s \wedge \neg fB \in s$$

$$\boldsymbol{evalcf\_ftv\_fb\_lemma} =$$
$$\vdash \forall s \bullet EvalCf\_ftv\ s = fB \Leftrightarrow \neg fFalse \in s \wedge \neg fT \in s \wedge fB \in s$$

$$\boldsymbol{evalcf\_ftv\_fb\_lemma1} =$$
$$\vdash \forall s \bullet EvalCf\_ftv\ s = fB \Rightarrow \neg fFalse \in s \wedge \neg fT \in s \wedge fB \in s$$

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

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

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

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

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

HOL Constant

$$
\begin{array}{l}
\boldsymbol{EvalFormFunct} : 't\ CFE \times 't\ REL \times ('a, 't)\ ST \to (GS \to ('a, 't)\ RV) \\
\quad\quad \to (GS \to ('a, 't)\ RV) \\
\hline
\forall cfe\ \$\leqslant_t\ st \bullet\ EvalFormFunct\ (cfe, \$\leqslant_t, st) = \lambda ef\ f \bullet \\
\quad\quad if\ f \in Syntax \\
\quad\quad then\ if\ IsAf\ f \\
\quad\quad\quad\quad then\ EvalAf\ \$\leqslant_t\ f\ st \\
\quad\quad\quad\quad else \\
\quad\quad\quad\quad\quad let\ rvs = FunImage\ ef\ (X_g(CfForms\ f)) \\
\quad\quad\quad\quad\quad in\ \ EvalCf\ cfe\ f\ st\ rvs \\
\quad\quad else\ \epsilon x \bullet T
\end{array}
$$

The semantics of formulae is then given by:

HOL Constant

$$
\begin{array}{l}
\boldsymbol{EvalForm} : 't\ CFE \times 't\ REL \times ('a, 't)\ ST \to GS \to ('a, 't)\ RV \\
\hline
\forall cfe\ \$\leqslant_t\ st \bullet\ EvalForm\ (cfe, \$\leqslant_t, st) = fix\ (EvalFormFunct\ (cfe, \$\leqslant_t, st))
\end{array}
$$

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.

$$
\begin{array}{l}
\boldsymbol{evalformfunct\_respect\_thm} = \\
\quad \vdash \forall\ cfe\ \leqslant_t\ st \bullet\ EvalFormFunct\ (cfe, \leqslant_t, st)\ respects\ ScPrec \\
\\
\boldsymbol{evalformfunct\_fixp\_lemma} = \\
\quad \vdash \forall\ cfe\ \leqslant_t\ st \\
\quad\quad \bullet\ EvalForm\ (cfe, \leqslant_t, st) \\
\quad\quad\quad = EvalFormFunct\ (cfe, \leqslant_t, st)\ (EvalForm\ (cfe, \leqslant_t, st))
\end{array}
$$

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

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

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

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

$$\textbf{EvalForm\_fT\_lemma2} =$$
$$\vdash \forall\ y$$
$$\bullet\ y \in Syntax$$
$$\Rightarrow (\forall\ va$$
$$\bullet\ FreeVars\ y \subseteq IxDom\ va$$
$$\wedge\ IxRan\ va \subseteq V \cup \{\varnothing_g\}$$
$$\wedge\ EvalForm\ (EvalCf\_ftv, \$\leqslant_{t4},\ V, \$\in_v)\ y\ va = fT$$
$$\Rightarrow (\exists\ x\ y\bullet\ x \in V \cup \{\varnothing_g\} \wedge y \in V \cup \{\varnothing_g\} \wedge x \in_v y = fT))$$

2.2.3 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

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

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

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

2.2.4 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

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

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

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

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

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

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

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

sto_lubs_exist_thm =

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

sto_glbs_exist_thm =

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

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, "≤ft3");
```

HOL Constant

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

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

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

Lets now get on with the four valued case.

SML

```
declare_infix(300, "⩽ft4");
```

HOL Constant

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

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

$$\leqslant_{\boldsymbol{ft4}}\boldsymbol{\_crpou\_thm} =$$
$$\vdash CRpoU\ \$\leqslant_{ft4}$$

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

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

$$\forall V\ r\bullet\ ExtSRO\ (V, r) = \lambda x\ y\bullet \forall z\bullet\ z \in V \Rightarrow r\ z\ x \leqslant_{t4} r\ z\ y$$

and to essences.

HOL Constant

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

$$\forall V\ r\bullet\ EssSRO\ (V, r) = \lambda x\ y\bullet \forall z\bullet\ z \in V \Rightarrow r\ x\ z \leqslant_{t4} r\ y\ z$$

Though I suspect I may only need:

HOL Constant

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

$$\forall s\bullet\ ExsSRO\ s = ConjOrder\ (ExtSRO\ s)\ (EssSRO\ s)$$

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

HOL Constant

$$\boldsymbol{ExsVaO} : (\prime a,\ GS)\ IX\ SET \times (GS\ SET \times (GS,\ FTV)BR) \to (\prime a,\ GS)\ IX \to (\prime a,\ GS)\ IX \to BOOL$$

$$\forall d\ s \bullet\ ExsVaO\ (d,\ s) = IxO2\ (d,\ (ExsSRO\ s))$$

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

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

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

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

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

2.2.5 Monotonicity Results

First we prove the monotonicity of *EvalForm*.

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

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

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

$$\boldsymbol{MonoEvalForm} : \prime t\ CFE \times \prime t\ REL \times \prime a\ SET \times GS \to (\prime a, \prime t)\ BR \to (\prime a, \prime t)\ RV$$

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

monoevalform_increasing_lemma =
⊢ ∀ c r s g
• CRpoU r ∧ Increasing (SetO r) r c
⇒ Increasing (Pw (Pw r)) (RvO r) (MonoEvalForm (c, r, s, g))

evalform_increasing_thm =
⊢ ∀ c r s g
• CRpoU r ∧ Increasing (SetO r) r c
⇒ Increasing (Pw (Pw r)) (RvO r) (λ ris• EvalForm (c, r, s, ris) g)

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.

evalform_increasing_thm2 =
⊢ ∀ (V, $∈$_v) g
• V ⊆ Syntax ∧ g ∈ Syntax
⇒ Increasing
(ExsVaO (V2IxSet V, V, $∈$_v))
$⩽$_t4
(EvalForm (EvalCf_ftv, $⩽$_t4, V, $∈$_v) g)

evalform_increasing_thm3 =
⊢ ∀ V $∈$_v g
• V ⊆ Syntax ∧ g ∈ Syntax
⇒ Increasing
(ExsVaO (V2IxSet (V ∪ {∅_g}), V ∪ {∅_g}, $∈$_v))
$⩽$_t4
(EvalForm (EvalCf_ftv, $⩽$_t4, V, $∈$_v) g)

evalform_increasing_thm4 =
⊢ ∀ V W $∈$_v g
• V ⊆ W ∧ V ⊆ Syntax ∧ g ∈ Syntax
⇒ Increasing
(ExsVaO (V2IxSet W, W, $∈$_v))
$⩽$_t4
(EvalForm (EvalCf_ftv, $⩽$_t4, V, $∈$_v) g)

2.2.6 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
Extension : (GS, FTV)BR → (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

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

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

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

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

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

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

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

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

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

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 [2].

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{CompExt}} : GS\ SET \rightarrow (GS, FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL \\
\hline
\forall V\ r \bullet\ CompExt\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ z\ x;\ r\ z\ y\} \in CompFTV
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{CoCompExt}} : GS\ SET \rightarrow (GS, FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL \\
\hline
\forall V\ r \bullet\ CoCompExt\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ z\ x;\ r\ z\ y\} \in CoCompFTV
\end{array}
$$

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

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{CompEss}} : GS\ SET \rightarrow (GS, FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL \\
\hline
\forall V\ r \bullet\ CompEss\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CompFTV
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{CoCompEss}} : GS\ SET \rightarrow (GS, FTV)BR \rightarrow GS \rightarrow GS \rightarrow BOOL \\
\hline
\forall V\ r \bullet\ CoCompEss\ V\ r = \lambda x\ y \bullet \forall z \bullet\ z \in V \Rightarrow \{r\ x\ z;\ r\ y\ z\} \in CoCompFTV
\end{array}
$$

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

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

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

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

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)
```

2.2.7 Proof Contexts

SML

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

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

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

force_new_pc "ifos1";
merge_pcs ["misc21", "'ifos"] "ifos1";
commit_pc "ifos1";
```

3 SEMANTIC FIXED POINTS

We now look for fixed points of the semantics of infinitary set theory.

SML

```
open_theory "ifos";
new_parent "membership";
force_new_theory "sfp";
force_new_pc "'sfp";
merge_pcs ["'savedthm_cs_∃_proof"] "'sfp";
set_merge_pcs ["misc21", "'ifos", "'sfp"];
```

3.1 The Semantic Functor

First we use the semantics above to define a functor which transforms membership relations.

The input membership relation is that in the membership structure relative to which the semantics is defined.

The output membership relation is obtained by considering the sets determined by formulae whose sole free variable is the empty set. Thus. under this resulting membership relation, (which will be a four-valued relation) the truth value of the claim that m is a member of s (where m and s are elements of the domain of discourse, which is sets coding formulae of infinitary first order set theory) is the value under the semantics for the formulae coded by s in the context of the variable assignment in which only the variable coded by the empty set is assigned a value and that value is t.

The following function creates a variable assignment with just a value for the empty set.

HOL Constant

$$\boldsymbol{Param\_\varnothing} : GS \to (GS,\ GS)\ IX$$

$$\forall p \bullet\ Param\_\varnothing\ p = \lambda x \bullet\ if\ x = \varnothing_g\ then\ Value\ p\ else\ Undefined$$

$$\boldsymbol{ixdom\_param\_\varnothing\_lemma} =$$
$$\vdash\ \forall\ x \bullet\ IxDom\ (Param\_\varnothing\ x) = \{\varnothing_g\}$$

$$\boldsymbol{ixran\_param\_\varnothing\_lemma} =$$
$$\vdash\ \forall\ x \bullet\ IxRan\ (Param\_\varnothing\ x) = \{x\}$$

$$\boldsymbol{ixval\_param\_\varnothing\_lemma} =$$
$$\vdash\ \forall\ x \bullet\ IxVal\ (Param\_\varnothing\ x)\ \varnothing_g = x$$

$$\boldsymbol{param\_\varnothing\_undefined\_lemma} =$$
$$\vdash\ \forall\ x\ y \bullet\ Param\_\varnothing\ x\ y = Undefined \Leftrightarrow \neg\ y = \varnothing_g$$

$$\textbf{\textit{Param\_}}\varnothing\textbf{\textit{\_Increasing\_lemma}} =$$
$$\vdash \forall\ V\ r$$
$$\bullet\ \mathit{Increasing}\ (V \lhd_o \mathit{ExsSRO}\ (V,\ r))\ (\mathit{ExsVaO}\ (\mathit{V2IxSet}\ V,\ V,\ r))\ \mathit{Param\_}\varnothing$$

$$\textbf{\textit{Param\_}}\varnothing\textbf{\textit{\_Increasing\_lemma2}} =$$
$$\vdash \forall\ V\ W\ r$$
$$\bullet\ \mathit{Increasing}\ (W \lhd_o \mathit{ExsSRO}\ (V,\ r))\ (\mathit{ExsVaO}\ (\mathit{V2IxSet}\ W,\ V,\ r))\ \mathit{Param\_}\varnothing$$

The "semantic functor", a parameterised partial relation transformation, is then defined thus:

HOL Constant

$$\textbf{\textit{Sf}}\colon \mathit{GS\ SET} \to (\mathit{GS},\ \mathit{FTV})\ \mathit{BR} \to (\mathit{GS},\ \mathit{FTV})\ \mathit{BR}$$

$$\forall\ d\ \$\in_v \bullet\ \mathit{Sf}\ d\ \$\in_v =$$
$$\lambda m\ s \bullet\ \mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ (d,\ \$\in_v))\ s\ (\mathit{Param\_}\varnothing\ m)$$

It is easily shown to be monotonic.

$$\textbf{\textit{sf\_increasing\_thm}} =$$
$$\vdash \forall\ d \bullet\ \mathit{Increasing}\ \$\leqslant_\in\ \$\leqslant_\in\ (\mathit{Sf}\ d)$$

We need to show that the essences of objects after application of the semantic functor depend only on their extension and essence before application of the functor. One way of expressing this is by stating that the identity function is an order morphism of between two distinct orderings of our syntax. The orderings are respectively:

- the ordering of syntactic objects according to their extension and essence relative to some membership relation $\$ \in_v$
- the ordering of the objects according to their essence in the relation obtained when the semantic functor is applied to $\$ \in_v$.

When the details are worked out we get:

$$\textbf{\textit{essence\_exstentional\_lemma}} =$$
$$\vdash \forall\ V\ W\ \$\in_v$$
$$\bullet\ V \subseteq W \wedge V \subseteq \mathit{Syntax}$$
$$\Rightarrow \mathit{Increasing}$$
$$(V \lhd_o \mathit{ExsSRO}\ (W,\ \$\in_v))$$
$$(\mathit{EssSRO}\ (V,\ \mathit{Sf}\ V\ \$\in_v))$$
$$\mathit{CombI}$$

$$\textbf{\textit{essence\_exstentional\_lemma2}} =$$
$$\vdash \forall\ V\ W\ \$\in_v$$
$$\bullet\ V \subseteq W \wedge V \subseteq \mathit{Syntax}$$
$$\Rightarrow \mathit{Increasing}$$
$$(W \lhd_o \mathit{ExsSRO}\ (W,\ \$\in_v))$$
$$(\mathit{EssSRO}\ (V,\ \mathit{Sf}\ V\ \$\in_v))$$
$$\mathit{CombI}$$

Note that the orders between the identity function is here proven to be a morphism are not restricted to the syntax of infinitary set theory, but cover the whole type GS, and this fact will be essential to some of the subsequent proofs, since we will make special use of $\ulcorner \varnothing_g \urcorner$ as an extra pseudo-set.

3.2 Properties of Fixed Points

The first property of interest concerns the conditions under which a fixed point of the semantic functor yields an extensional model of set theory.

I suppose we might start by saying what a fixed point is:

HOL Constant

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

$$\forall d\ r\bullet\ SFFixp\ d\ r \Leftrightarrow Sf\ d\ r = r$$

I am interested in interpretations of classical two-valued set theory and therefore will consider mainly fixed points which are total over the domain.

HOL Constant

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

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

$$\boldsymbol{totover\_lemma} =$$
$$\vdash \forall\ V\ r$$
$$\bullet\ TotalOver\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow r\ x\ y = fTrue \vee r\ x\ y = fFalse)$$

$$\boldsymbol{totover\_true\_lemma} =$$
$$\vdash \forall\ V\ r$$
$$\bullet\ TotalOver\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (fTrue \leqslant_{t4} r\ x\ y \Leftrightarrow r\ x\ y = fTrue))$$

$$\boldsymbol{totover\_false\_lemma} =$$
$$\vdash \forall\ V\ r$$
$$\bullet\ TotalOver\ V\ r$$
$$\Rightarrow (\forall\ x\ y\bullet\ x \in V \wedge y \in V \Rightarrow (fFalse \leqslant_{t4} r\ x\ y \Leftrightarrow r\ x\ y = fFalse))$$

If a relationship is total over some domain we can convert it to a boolean relation, without loss of information (provided we remember the domain).

HOL Constant

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

$$\forall\ r{:}('a,\ FTV)BR\bullet\ BoolRel\ r = \lambda x\ y\bullet\ fTrue \leqslant_{t4} r\ x\ y$$

Having chosen representatives for sets which are not canonical, we will be taking a quotient to eliminate redundancy. Two complementary concepts are useful in this, the *extension* of a set (which is the collection of its members) and the *essence* of a set (which is the collection of things of which it is a member).

These concepts don't work out very tidily when we are dealing with partial membership relationships. It seems easier not to formally define them but to define the relationships between two representatives whose extensions and essences are the same.

We now provide two ways of defining an equality relation over a boolean membership relationship.

The first is the obvious definition for set theoretic equality, two sets being equal if they have the same extension.

HOL Constant

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

$$\forall V\ r\bullet\ BEqRel\ V\ r = \lambda x\ y\bullet\ \forall z\bullet\ z \in V \Rightarrow (r\ z\ x \Leftrightarrow r\ z\ y)$$

The second is a stricter definition, two sets are equivalent if they have the same extension and the same essence,

HOL Constant

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

$$\forall V\ r\bullet\ BEqqRel\ V\ r = \lambda x\ y\bullet\ \forall z\bullet\ z \in V \Rightarrow (r\ z\ x \Leftrightarrow r\ z\ y) \wedge (r\ x\ z \Leftrightarrow r\ y\ z)$$

$$\boldsymbol{BEqqRel\_exs\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x\ y\bullet\ x \in V\ \wedge\ y \in V\ \wedge\ BEqqRel\ V\ r\ x\ y \Rightarrow r\ x\ x = r\ y\ y\ \wedge\ r\ x\ y = r\ y\ x$$

HOL Constant

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

$$\forall V\ r\bullet\ BoolEqRel\ V\ r = BEqRel\ V\ (BoolRel\ r)$$

HOL Constant

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

$$\forall V\ r\bullet\ BoolEqqRel\ V\ r = BEqqRel\ V\ (BoolRel\ r)$$

$$\boldsymbol{beqrel\_equiv\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ BEqRel\ V\ r)$$

$$\boldsymbol{beqqrel\_equiv\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ BEqqRel\ V\ r)$$

$$\boldsymbol{booleqrel\_equiv\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ BoolEqRel\ V\ r)$$

$$\boldsymbol{booleqqrel\_equiv\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ Equiv\ (V,\ BoolEqqRel\ V\ r)$$

The notation for sets we are using here is not canonical so each set has multiple representatives and we do not have extensionality. However, we can hope to be able to obtain an extensional interpretation by taking a quotient relative to extensional equality. The following condition suffices for this to be possible.

What this says is that whenever two sets are not the members of all the same sets then they do not have the same members.

HOL Constant

$$\textbf{\textit{PreExtensional}} : GS\ SET \to (GS,\ FTV)BR \to BOOL$$

$$\begin{aligned}&\forall d\ r \bullet PreExtensional\ d\ r \Leftrightarrow \neg\ OverDefined\ d\ r \wedge \forall x\ y \bullet x \in d \wedge y \in d\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ x\ z;\ r\ y\ z\}\ fT)\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ z\ x;\ r\ z\ y\}\ fT)\end{aligned}$$

However, that turns out a tad too strong (we need this to be a sufficient property to ensure that a total least fixed point is extensional, but it must be preserved by the semantic functor, and that one probably isn't).

HOL Constant

$$\textbf{\textit{PreExtensional2}} : GS\ SET \to (GS,\ FTV)BR \to BOOL$$

$$\begin{aligned}&\forall d\ r \bullet PreExtensional2\ d\ r \Leftrightarrow \neg\ OverDefined\ d\ r \wedge \forall x\ y \bullet x \in d \wedge y \in d\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ x\ z;\ r\ y\ z\}\ fT)\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ z\ x;\ r\ z\ y\}\ fT)\\ &\qquad \vee\ Lub\ \$\leqslant_{t4}\ \{r\ x\ x;\ r\ y\ y\} = fT\\ &\qquad \vee\ Lub\ \$\leqslant_{t4}\ \{r\ x\ y;\ r\ y\ x\} = fT\end{aligned}$$

HOL Constant

$$\textbf{\textit{PreExtensional3}} : GS\ SET \to (GS,\ FTV)BR \to BOOL$$

$$\begin{aligned}&\forall d\ r \bullet PreExtensional3\ d\ r \Leftrightarrow \neg\ OverDefined\ d\ r \wedge \forall x\ y \bullet x \in d \wedge y \in d\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge \neg\ \{r\ x\ z;\ r\ y\ z\} \in CompFTV)\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge \neg\ \{r\ z\ x;\ r\ z\ y\} \in CompFTV)\\ &\qquad \vee\ \neg\ \{r\ x\ x;\ r\ y\ y\} \in CompFTV\\ &\qquad \vee\ \neg\ \{r\ x\ y;\ r\ y\ x\} \in CompFTV\end{aligned}$$

It is useful in the present context to have the following result:

There is a corresponding property over the boolean version of a total partial relation.

HOL Constant

$$\textbf{\textit{TotExtensional}} : GS\ SET \to (GS,\ BOOL)BR \to BOOL$$

$$\begin{aligned}&\forall d\ r \bullet TotExtensional\ d\ r \Leftrightarrow \forall x\ y \bullet x \in d \wedge y \in d\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge \neg(r\ x\ z \Leftrightarrow r\ y\ z))\\ &\qquad \Rightarrow (\exists z \bullet z \in d \wedge \neg(r\ z\ x \Leftrightarrow r\ z\ y))\end{aligned}$$

and a weakened version:

HOL Constant

$$\boldsymbol{TotExtensional2} : GS\ SET \to (GS,\ BOOL)BR \to BOOL$$

$$\forall d\ r\bullet\ TotExtensional2\ d\ r \Leftrightarrow \forall x\ y\bullet\ x \in d \wedge y \in d \Rightarrow (\exists z\bullet\ z \in d \wedge \neg(r\ x\ z \Leftrightarrow r\ y\ z)) \Rightarrow (\exists z\bullet\ z \in d \wedge \neg(r\ z\ x \Leftrightarrow r\ z\ y)) \vee \neg(r\ x\ x \Leftrightarrow r\ y\ y) \vee \neg(r\ x\ y \Leftrightarrow r\ y\ x)$$

$$\boldsymbol{pre\_tot\_ext\_lemma} =$$
$$\vdash \forall\ V\ pr \bullet\ PreExtensional\ V\ pr \wedge TotalOver\ V\ pr \Rightarrow TotExtensional\ V\ (BoolRel\ pr)$$

$$\boldsymbol{pre2\_tot\_ext\_lemma} =$$
$$\vdash \forall\ V\ pr \bullet\ PreExtensional2\ V\ pr \wedge TotalOver\ V\ pr \Rightarrow TotExtensional2\ V\ (BoolRel\ pr)$$

If a membership relation is total and pre-extensional over some domain, then we can obtain from it an extensional boolean membership structure whose domain is equivalence classes under co-extensionality of the original domain.

I will be looking to establish reasonable conditions under which the fixed points of the semantic functor will be *PreExtensional*, but first I need to establish that *PreExtensional*ity suffices for a fixedpoint of the semantic functor to yield and extensional interpretation of first order set theory.

My first shot at constructing a structure from a total fixed point used co-extensionality as equality, which is OK provided that the fixed point is pre-extensional, in which case the resulting structure is extensional. This is it:

HOL Constant

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

$$\forall\ (V{:}GS\ SET)\ (r{:}(GS,\ FTV)\ BR)\bullet\ MSfromSFF\ (V,\ r) = \text{let } \$\in_v = BoolRel\ r \text{ and } \$=_v = BEqRel\ V\ (BoolRel\ r) \text{ in } (QuotientSet\ V\ \$=_v,\ \lambda s\ t\bullet\ \forall x\ y\bullet\ x \in s \wedge y \in t \Rightarrow x \in_v y)$$

However, it now seems to me better to define the structure with an equality relation which is stricter if the membership relation is not pre-extensional. The result is the same if it is pre-extensional (conjecture) but the advantage is that if it is not we still get a sensible membership structure (though not an extensional one).

Put another way, this corrects a defect in the first version, that it doesn't work for non-extensional fixed points. In order to get a good structure from a fixed point which is not pre-extensional, you have to regard two sets as distinct if they are not members of the same sets, even if they have the same extension. The result is that the axiom of extensionality will not hold unqualified. Admitting this possibility is a good idea since the kind of type structure I have in mind will probably be a model for stratified comprehension, and may not be compatible with full extensionality.

HOL Constant

$$\boldsymbol{MSfromSFF2} : GS\ SET \times (GS,\ FTV)\ BR \to GS\ SET\ SET \times (GS\ SET,\ BOOL)\ BR$$

$$\forall\ (V{:}GS\ SET)\ (r{:}(GS,\ FTV)\ BR)\bullet\ MSfromSFF2\ (V,\ r) =$$
$$\quad let\ \$\in_v\ =\ BoolRel\ r$$
$$\quad and\ \$=_v\ =\ BEqqRel\ V\ (BoolRel\ r)$$
$$\quad in\ (QuotientSet\ V\ \$=_v,\ \lambda s\ t\bullet\ \forall x\ y\bullet\ x \in s\ \wedge\ y \in t \Rightarrow x \in_v y)$$

$$\boldsymbol{equivclass\_mem\_lemma} =$$
$$\vdash\ \forall\ V\ r\ x$$
$$\bullet\ x \in EquivClass\ (V,\ BEqRel\ V\ (BoolRel\ r))\ z$$
$$\Leftrightarrow x \in V\ \wedge\ (\forall\ y\bullet\ y \in V \Rightarrow (BoolRel\ r\ y\ x \Leftrightarrow BoolRel\ r\ y\ z))$$

$$\boldsymbol{equivclass\_mem\_lemma2} =$$
$$\vdash\ \forall\ V\ r\ x\bullet\ x \in EquivClass\ (V,\ BEqqRel\ V\ (BoolRel\ r))\ z$$
$$\Leftrightarrow x \in V\ \wedge\ (\forall\ y\bullet\ y \in V$$
$$\Rightarrow (BoolRel\ r\ y\ x \Leftrightarrow BoolRel\ r\ y\ z)$$
$$\wedge\ (BoolRel\ r\ x\ y \Leftrightarrow BoolRel\ r\ z\ y))$$

$$\boldsymbol{rep\_independence\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ TotalOver\ V\ r \Rightarrow (\forall\ s\ t\bullet\ s \in V\ \wedge\ t \in V \Rightarrow$$
$$(let\ u\ =\ EquivClass\ (V,\ BoolEqqRel\ V\ r)\ s$$
$$and\ v\ =\ EquivClass\ (V,\ BoolEqqRel\ V\ r)\ t$$
$$in\ (\forall\ x\ y\bullet\ x \in u\ \wedge\ y \in v \Rightarrow BoolRel\ r\ x\ y)$$
$$\Leftrightarrow (\exists\ x\ y\bullet\ x \in u\ \wedge\ y \in v\ \wedge\ BoolRel\ r\ x\ y)))$$

$$\boldsymbol{totpre\_rep\_independence\_lemma} =$$
$$\vdash\ \forall\ V\ r\bullet\ TotalOver\ V\ r\ \wedge\ PreExtensional\ V\ r$$
$$\Rightarrow (\forall\ s\ t\bullet\ s \in V\ \wedge\ t \in V$$
$$\Rightarrow (let\ u\ =\ EquivClass\ (V,\ BoolEqRel\ V\ r)\ s$$
$$and\ v\ =\ EquivClass\ (V,\ BoolEqRel\ V\ r)\ t$$
$$in\ (\forall\ x\ y\bullet\ x \in u\ \wedge\ y \in v \Rightarrow BoolRel\ r\ x\ y)$$
$$\Leftrightarrow (\exists\ x\ y\bullet\ x \in u\ \wedge\ y \in v\ \wedge\ BoolRel\ r\ x\ y)))$$

$$\boldsymbol{totpre\_rep\_independence\_lemma2} =$$
$$\vdash\ \forall\ V\ r\bullet\ TotalOver\ V\ r\ \wedge\ PreExtensional\ V\ r$$
$$\Rightarrow (\forall\ s\ t\ u\ v$$
$$\bullet\ s \in V$$
$$\wedge\ t \in V$$
$$\wedge\ u \in V$$
$$\wedge\ v \in V$$
$$\wedge\ BoolEqRel\ V\ r\ s\ u$$
$$\wedge\ BoolEqRel\ V\ r\ t\ v$$
$$\Rightarrow (BoolRel\ r\ s\ t \Leftrightarrow BoolRel\ r\ u\ v))$$

$$
\begin{array}{l}
\textbf{\textit{rep\_independence\_lemma2}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ \mathit{TotalOver}\ V\ r \\
\quad\quad\quad \Rightarrow (\forall\ s\ t\ u\ v \\
\quad\quad\quad \bullet\ s \in V \\
\quad\quad\quad\quad\quad \wedge\ t \in V \\
\quad\quad\quad\quad\quad \wedge\ u \in V \\
\quad\quad\quad\quad\quad \wedge\ v \in V \\
\quad\quad\quad\quad\quad \wedge\ \mathit{BoolEqqRel}\ V\ r\ s\ u \\
\quad\quad\quad\quad\quad \wedge\ \mathit{BoolEqqRel}\ V\ r\ t\ v \\
\quad\quad\quad\quad\quad \Rightarrow (\mathit{BoolRel}\ r\ s\ t \Leftrightarrow \mathit{BoolRel}\ r\ u\ v))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{eq\_eq\_eqq\_lemma}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ \mathit{TotalOver}\ V\ r \wedge \mathit{PreExtensional}\ V\ r \\
\quad\quad\quad \Rightarrow (\forall\ x\ y \\
\quad\quad\quad \bullet\ x \in V \wedge y \in V \\
\quad\quad\quad\quad \Rightarrow (\mathit{BEqRel}\ V\ (\mathit{BoolRel}\ r)\ x\ y \Leftrightarrow \mathit{BEqqRel}\ V\ (\mathit{BoolRel}\ r)\ x\ y))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{msfromsff\_eq\_lemma}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ \mathit{TotalOver}\ V\ r \wedge \mathit{PreExtensional}\ V\ r \\
\quad\quad\quad \Rightarrow \mathit{MSfromSFF}\ (V,\ r) = \mathit{MSfromSFF2}\ (V,\ r)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{preext\_ext\_lemma}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ \mathit{TotalOver}\ V\ r \wedge \mathit{PreExtensional}\ V\ r \Rightarrow \mathit{extensional}\ (\mathit{MSfromSFF}\ (V,\ r))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{preext\_ext\_lemma2}} = \\
\quad \vdash \forall\ V\ r \\
\quad\quad \bullet\ \mathit{TotalOver}\ V\ r \wedge \mathit{PreExtensional}\ V\ r \Rightarrow \mathit{extensional}\ (\mathit{MSfromSFF2}\ (V,\ r))
\end{array}
$$

3.3 Properties of the Semantic Functor

I hope to prove that least fixed points of the semantic functor are always pre-extensional. If they are also total the resulting interpretation will be extensional. I'm also interested in the dual of this result, for greatest fixed points.

A semantic functor is extensional over some domain if any two sets which have the same extension and the same essence over that domain in the input relation have the same essence in the result of applying the functor.

HOL Constant

$$\textbf{\textit{ExtSem}} : GS\ SET \to (GS\ SET \to (GS,\ FTV)BR \to (GS,\ FTV)BR) \to BOOL$$

$$\forall V\ f\bullet\ ExtSem\ V\ f \Leftrightarrow \forall r\bullet\ \forall x\ y\bullet\ x \in V \wedge y \in V$$
$$\wedge\ SameExt\ V\ r\ x\ y \wedge SameEss\ V\ r\ x\ y$$
$$\Rightarrow SameEss\ V\ (f\ V\ r)\ x\ y$$

$$\textbf{\textit{extsem\_sf\_thm}} =$$
$$\vdash\ \forall\ V\bullet\ V \subseteq SetReps \Rightarrow ExtSem\ V\ Sf$$

ExtSem turns out to be too weak. The following works with compatibility of extension and essence rather than identity.

HOL Constant

$$\textbf{\textit{CompExtSem}} : GS\ SET \to (GS\ SET \to (GS,\ FTV)BR \to (GS,\ FTV)BR) \to BOOL$$

$$\forall V\ f\bullet\ CompExtSem\ V\ f \Leftrightarrow \forall r\bullet \neg\ OverDefined\ V\ r \Rightarrow$$
$$\forall x\ y\bullet\ x \in V \wedge y \in V \wedge Compatible\ V\ r\ x\ y$$
$$\Rightarrow CompEss\ V\ (f\ V\ r)\ x\ y$$

That too is not enough. This one might do the trick.

HOL Constant

$$\textbf{\textit{CompExtSem2}} : GS\ SET \to (GS\ SET \to (GS,\ FTV)BR \to (GS,\ FTV)BR) \to BOOL$$

$$\forall V\ f\bullet\ CompExtSem2\ V\ f \Leftrightarrow \forall r\bullet \neg\ OverDefined\ V\ r \Rightarrow$$
$$\forall x\ y\bullet\ x \in V \wedge y \in V \wedge Compatible\ V\ r\ x\ y$$
$$\Rightarrow CompEss\ V\ (f\ V\ r)\ x\ y \wedge \{r\ x\ x;\ r\ x\ y;\ r\ y\ x;\ r\ y\ y\} \in CompFTV$$

To prove that the semantic functor has this property is a bit harder than the proof for *ExtSem*.

3.3.1 Extending Partial Membership Structures

The idea of compatibility is that from what we know two elements could turn out to have the same extension or essence when all undefined values are refined to definite values by repeated application of the semantic functor. Hence, if two elements are compatible (and not overdefined) then there is some other possible extension or essence which refines both (and is not overdefined). If we apply the semantic functor to something having that extension or essence then, because the semantic functor is monotonic and the computation of essence is extensional (depends on the extension and intension of the elements but does not refer to their syntactic structure), we will get something whose essence is an upper bound of the essences of the two compatible elements. Hence the two elements have compatible essences after the application of the semantic functor.

There is an awkwardness in this plan because the semantic functor applies to a partial membership relationship, not to arbitrary essences and extensions. It may be that it will suffice to talk hypothetically. "if there was an element which was a not overdefined upper bound for the an arbitrary pair of compatible element then its extension after application of the semantic functor would be an upper bound for the extensions of the two elements in the image of the semantic functor, which would suffice to show that they are compatible".

To carry through this plan we need a lemma to the effect that compatibility of not overdefined elements is equivalent to the existence of a not overdefined upper bound for the elements. We do this in two parts, showing the existence separately of not overdefined upper bounds on extensions and essences. It looks like there will be *least* upper bounds, but I don't think that I need to prove the stronger result.

$$
\begin{array}{l}
\textbf{\textit{compext\_lemma1}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad \Rightarrow (\forall\ r\ x\ y \\
\quad\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad\quad \Rightarrow (\mathit{CompExt}\ V\ r\ x\ y \\
\quad\quad\quad\quad\quad\quad \Leftrightarrow (\exists\ xt \\
\quad\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsUb} \\
\quad\quad\quad\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad\quad\quad\quad \{\mathit{Extension}\ r\ x;\ \mathit{Extension}\ r\ y\} \\
\quad\quad\quad\quad\quad\quad\quad\quad xt)))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{compext\_lemma1b}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ r\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad \Rightarrow (\mathit{CompExt}\ V\ r\ x\ y \\
\quad\quad\quad\quad\quad \Leftrightarrow (\exists\ xt \\
\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsUb}\ (\mathit{PwS}\ V\ \$\leqslant_{t4})\ \{\mathit{Extension}\ r\ x;\ \mathit{Extension}\ r\ y\}\ xt)))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{compext\_lemma2}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad \Rightarrow (\forall\ \$\in_v\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad \Rightarrow \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad\quad = (\exists\ xt \\
\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsLub} \\
\quad\quad\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad\quad\quad \{\mathit{Extension}\ \$\in_v\ x;\ \mathit{Extension}\ \$\in_v\ y\} \\
\quad\quad\quad\quad\quad\quad\quad xt))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{compext\_lemma2b}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ \$\in_v\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad \Rightarrow (\mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad\quad \Leftrightarrow (\exists\ xt \\
\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsLub} \\
\quad\quad\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad\quad\quad \{\mathit{Extension}\ \$\in_v\ x;\ \mathit{Extension}\ \$\in_v\ y\} \\
\quad\quad\quad\quad\quad\quad\quad xt)))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{compess\_lemma1}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad \Rightarrow (\forall\ r\ x\ y \\
\quad\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad\quad \Rightarrow (\mathit{CompEss}\ V\ r\ x\ y \\
\quad\quad\quad\quad\quad\quad \Leftrightarrow (\exists\ es \\
\quad\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ es \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsUb} \\
\quad\quad\quad\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad\quad\quad\quad \{\mathit{Essence}\ r\ x;\ \mathit{Essence}\ r\ y\} \\
\quad\quad\quad\quad\quad\quad\quad\quad es)))
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{compess\_lemma1b}} = \\
\quad \vdash \forall\ V \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad \Rightarrow (\forall\ r\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V \\
\quad\quad\quad\quad \Rightarrow (\mathit{CompEss}\ V\ r\ x\ y \\
\quad\quad\quad\quad\quad \Leftrightarrow (\exists\ es \\
\quad\quad\quad\quad\quad \bullet\ \neg\ \mathit{OverDefinedL}\ V\ es \\
\quad\quad\quad\quad\quad\quad \wedge\ \mathit{IsUb}\ (\mathit{PwS}\ V\ \$\leqslant_{t4})\ \{\mathit{Essence}\ r\ x;\ \mathit{Essence}\ r\ y\}\ es)))
\end{array}
$$

```
compess_lemma2 =
  ⊢ ∀ V
    • V ⊆ SetReps
        ⇒ (∀ $∈_v x y
        • x ∈ V ∧ y ∈ V
            ⇒ CompEss V $∈_v x y
              = (∃ es
              • ¬ OverDefinedL V es
                  ∧ IsLub
                    (PwS V $⩽_t4)
                    {Essence $∈_v x; Essence $∈_v y}
                    es))
```

```
compess_lemma2b =
  ⊢ ∀ V
    • V ⊆ Syntax
        ⇒ (∀ $∈_v x y
        • x ∈ V ∧ y ∈ V
            ⇒ (CompEss V $∈_v x y
              ⇔ (∃ es
              • ¬ OverDefinedL V es
                  ∧ IsLub
                    (PwS V $⩽_t4)
                    {Essence $∈_v x; Essence $∈_v y}
                    es)))
```

If there were a setrep z whose extension and essence were upper bounds for two setsreps x and y which have compatible and not overdefined extension and essences, then we could use the image of z under the semantics to show that the imagss of x and y have compatible essences. Unfortunately there need not be any such setrep.

The proof plan, for showing that the compatible elements x and y will map to elements whose essences are compatible, is to add an extra element into the interpretation we are working with which has the required extensions and essences (not overdefined upper bounds of those of x and y).

In order to define the required extension I first use the above two lemmas to define (loosely) functions which yield not-overdefined upper bounds for extensions and essences.

HOL Constant

$$\textbf{\textit{ExtUb}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ xt = ExtUb\ (V,\ \$\in_v)\ x\ y \\
\quad in\ \neg\ OverDefinedL\ V\ xt \\
\qquad\qquad \wedge\ IsUb \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\} \\
\qquad\qquad\quad xt
\end{array}$$

HOL Constant

$$\textbf{\textit{ExtUb2}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ xt = ExtUb2\ (V,\ \$\in_v)\ x\ y \\
\quad in\ \neg\ OverDefinedL\ V\ xt \\
\qquad\qquad \wedge\ IsUb \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\} \\
\qquad\qquad\quad xt
\end{array}$$

HOL Constant

$$\textbf{\textit{ExtLub}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ xt = ExtLub\ (V,\ \$\in_v)\ x\ y \\
\quad in\ \neg\ OverDefinedL\ V\ xt \\
\qquad\qquad \wedge\ IsLub \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\} \\
\qquad\qquad\quad xt
\end{array}$$

HOL Constant

$$\textbf{\textit{ExtLub2}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ xt = ExtLub2\ (V,\ \$\in_v)\ x\ y \\
\quad in \neg\ OverDefinedL\ V\ xt \\
\qquad\qquad \wedge\ IsLub \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\} \\
\qquad\qquad\quad xt
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{ExtUb_lemma1}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\ \bullet\ V \subseteq SetReps \\
\qquad\quad \wedge\ x \in V \\
\qquad\quad \wedge\ y \in V \\
\qquad\quad \wedge\ CompExt\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow \neg\ OverDefinedL\ V\ (ExtUb\ (V,\ \$\in_v)\ x\ y) \\
\qquad\quad \wedge\ (\forall\ z\bullet\ z \in V \Rightarrow (z \in_v x) \leqslant_{t4} ExtUb\ (V,\ \$\in_v)\ x\ y\ z \\
\qquad\qquad \wedge\ (z \in_v y) \leqslant_{t4} ExtUb\ (V,\ \$\in_v)\ x\ y\ z)
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{ExtLub_lemma1}} = \\
\quad \vdash \forall(V,\ \$\in_v)\ x\ y\bullet \\
\qquad V \subseteq SetReps \wedge \neg\ OverDefined\ V\ \$\in_v \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \Rightarrow \\
\qquad \neg\ OverDefinedL\ V\ (ExtLub\ (V,\ \$\in_v)\ x\ y) \\
\qquad \wedge\ \forall z\bullet\ z \in V \Rightarrow (z \in_v x) \leqslant_{t4} ExtLub\ (V,\ \$\in_v)\ x\ y\ z \\
\qquad\quad \wedge\ (z \in_v y) \leqslant_{t4} ExtLub\ (V,\ \$\in_v)\ x\ y\ z
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{ExtLub2_lemma1}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\ \bullet\ V \subseteq Syntax \wedge \neg\ OverDefined\ V\ \$\in_v \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow \neg\ OverDefinedL\ V\ (ExtLub2\ (V,\ \$\in_v)\ x\ y) \\
\qquad\quad \wedge\ (\forall\ z \\
\qquad\quad \bullet\ z \in V \\
\qquad\qquad \Rightarrow (z \in_v x) \leqslant_{t4} ExtLub2\ (V,\ \$\in_v)\ x\ y\ z \\
\qquad\qquad\quad \wedge\ (z \in_v y) \leqslant_{t4} ExtLub2\ (V,\ \$\in_v)\ x\ y\ z)
\end{array}$$

HOL Constant

$$\textbf{\textit{EssUb}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ es = EssUb\ (V,\ \$\in_v)\ x\ y \\
\quad in \neg\ OverDefinedL\ V\ es \\
\qquad\qquad \wedge\ IsUb \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\} \\
\qquad\qquad\quad es
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{EssUb_lemma1}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\ \bullet\ V \subseteq SetReps \\
\qquad\quad \wedge\ x \in V \\
\qquad\quad \wedge\ y \in V \\
\qquad\quad \wedge\ CompEss\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow \neg\ OverDefinedL\ V\ (EssUb\ (V,\ \$\in_v)\ x\ y) \\
\qquad\quad \wedge\ (\forall\ z \\
\qquad\quad \bullet\ z \in V \\
\qquad\qquad \Rightarrow (x \in_v z) \leqslant_{t4} EssUb\ (V,\ \$\in_v)\ x\ y\ z \\
\qquad\qquad\quad \wedge\ (y \in_v z) \leqslant_{t4} EssUb\ (V,\ \$\in_v)\ x\ y\ z)
\end{array}$$

HOL Constant

$$\textbf{\textit{EssLub}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ es = EssLub\ (V,\ \$\in_v)\ x\ y \\
\quad in \neg\ OverDefinedL\ V\ es \\
\qquad\qquad \wedge\ IsLub \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\} \\
\qquad\qquad\quad es
\end{array}$$

HOL Constant

$$\textbf{\textit{EssLub2}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS \to BOOL\ DPO)$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad let\ es = EssLub2\ (V,\ \$\in_v)\ x\ y \\
\quad in \neg\ OverDefinedL\ V\ es \\
\qquad\qquad \wedge\ IsLub \\
\qquad\qquad\quad (PwS\ V\ \$\leqslant_{t4}) \\
\qquad\qquad\quad \{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\} \\
\qquad\qquad\quad es
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{EssLub_lemma1}} = \\
\quad \vdash \forall(V,\ \$\in_v)\ x\ y\bullet \\
\quad\quad V \subseteq SetReps \wedge \neg\ OverDefined\ V\ \$\in_v \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y \Rightarrow \\
\quad\quad \neg\ OverDefinedL\ V\ (EssLub\ (V,\ \$\in_v)\ x\ y) \\
\quad\quad \wedge\ \forall z\bullet\ z \in V \Rightarrow (x \in_v z) \leqslant_{t4} EssLub\ (V,\ \$\in_v)\ x\ y\ z \\
\quad\quad\quad\quad \wedge\ (y \in_v z) \leqslant_{t4} EssLub\ (V,\ \$\in_v)\ x\ y\ z
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{EssLub_lemma1}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq Syntax \wedge \neg\ OverDefined\ V\ \$\in_v \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y \\
\quad\quad\quad \Rightarrow \neg\ OverDefinedL\ V\ (EssLub2\ (V,\ \$\in_v)\ x\ y) \\
\quad\quad\quad\quad \wedge\ (\forall\ z \\
\quad\quad\quad\quad \bullet\ z \in V \\
\quad\quad\quad\quad\quad\quad \Rightarrow (x \in_v z) \leqslant_{t4} EssLub2\ (V,\ \$\in_v)\ x\ y\ z \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ (y \in_v z) \leqslant_{t4} EssLub2\ (V,\ \$\in_v)\ x\ y\ z)
\end{array}$$

The required extension to the membership relation is defined as follows:

HOL Constant

$$\textbf{\textit{MsExtend}} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS,\ FTV)BR$$

$$\begin{array}{l}
\forall(V,\ \$\in_v)\ x\ y\bullet\ MsExtend\ (V,\ \$\in_v)\ x\ y = \\
\quad let\ xt = ExtUb\ (V,\ \$\in_v)\ x\ y \\
\quad and\ es = EssUb\ (V,\ \$\in_v)\ x\ y \\
\quad in\ (\lambda x'\ y'\bullet \\
\qquad\quad if\ x' = \varnothing_g \\
\qquad\quad then\ if\ y' = \varnothing_g\ then\ Lub\ \$\leqslant_{t4}\ \{x \in_v x;\ y \in_v x;\ x \in_v y;\ y \in_v y\}\ else\ es\ y' \\
\qquad\quad else\ if\ y' = \varnothing_g\ then\ xt\ x'\ else\ x' \in_v y')
\end{array}$$

HOL Constant

$$\boldsymbol{MsExtend2} : GS\ SET \times (GS,\ FTV)BR \to GS \to GS \to (GS,\ FTV)BR$$

$$\forall(V,\ \$\in_v)\ x\ y\bullet\ MsExtend2\ (V,\ \$\in_v)\ x\ y =$$
$$\quad let\ xt = ExtLub2\ (V,\ \$\in_v)\ x\ y$$
$$\quad and\ es = EssLub2\ (V,\ \$\in_v)\ x\ y$$
$$\quad in\ (\lambda x'\ y'\bullet$$
$$\qquad if\ x' = \varnothing_g$$
$$\qquad then\ if\ y' = \varnothing_g\ then\ Lub\ \$\leqslant_{t4}\ \{x \in_v x;\ y \in_v x;\ x \in_v y;\ y \in_v y\}\ else\ es\ y'$$
$$\qquad else\ if\ y' = \varnothing_g\ then\ xt\ x'\ else\ x' \in_v y')$$

3.3.2 Properties of Extensions

First that these extensions are the same over the original domain as the unextended relation.

$$\boldsymbol{PmrEq\_MsExtend\_lemma1} =$$
$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\bullet\ V \subseteq Syntax \Rightarrow PmrEq\ V\ r\ (MsExtend\ (V,\ r)\ x\ y)$$

$$\boldsymbol{PmrEq\_MsExtend\_lemma2} =$$
$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\bullet\ V \subseteq Syntax \Rightarrow PmrEq\ V\ (MsExtend\ (V,\ r)\ x\ y)\ r$$

$$\boldsymbol{PmrEq\_MsExtend2\_lemma2} =$$
$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\bullet\ V \subseteq Syntax \Rightarrow PmrEq\ V\ (MsExtend2\ (V,\ r)\ x\ y)\ r$$

The evaluation of formulae using the extended relation gives the same results if the stiplulated domain of quantification remains the same. (I don't know why I did not obtain more general results than these, though these particular ones suffice to show that the resulting essences of the two elements determining an extension, after application of the semantic functor, are unaffected by the extension to the structure.)

$$\boldsymbol{EvalForm\_MsExtend\_lemma1} =$$
$$\vdash \forall\ (V,\ r)\ x\ y\ z$$
$$\quad \bullet\ V \subseteq Syntax \wedge y \in V \wedge z \in V$$
$$\qquad \Rightarrow EvalForm$$
$$\qquad\quad (EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ MsExtend\ (V,\ r)\ x\ y)$$
$$\qquad\quad z$$
$$\qquad\quad (Param\_\varnothing\ y)$$
$$\qquad\quad = EvalForm\ (EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ r)\ z\ (Param\_\varnothing\ y)$$

$$\boldsymbol{EvalForm\_MsExtend2\_lemma1} =$$
$$\vdash \forall\ (V,\ r)\ x\ y\ z$$
$$\quad \bullet\ V \subseteq Syntax \wedge y \in V \wedge z \in V$$
$$\qquad \Rightarrow EvalForm$$
$$\qquad\quad (EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ MsExtend2\ (V,\ r)\ x\ y)$$
$$\qquad\quad z$$
$$\qquad\quad (Param\_\varnothing\ y)$$
$$\qquad\quad = EvalForm\ (EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ r)\ z\ (Param\_\varnothing\ y)$$

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_MsExtend\_lemma2}} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow \mathit{EvalForm} \\
\quad\quad\quad\quad (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend}\ (V,\ r)\ x\ y) \\
\quad\quad\quad\quad z \\
\quad\quad\quad\quad (\mathit{Param\_}\varnothing\ x) \\
\quad\quad\quad\quad = \mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)\ z\ (\mathit{Param\_}\varnothing\ x)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_MsExtend2\_lemma2}} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow \mathit{EvalForm} \\
\quad\quad\quad\quad (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y) \\
\quad\quad\quad\quad z \\
\quad\quad\quad\quad (\mathit{Param\_}\varnothing\ x) \\
\quad\quad\quad\quad = \mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)\ z\ (\mathit{Param\_}\varnothing\ x)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{EvalForm\_MsExtend2\_lemma3}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y\ v\ w \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge y \in V \wedge v \in V \wedge w \in V \\
\quad\quad\quad \Rightarrow \mathit{EvalForm} \\
\quad\quad\quad\quad (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y) \\
\quad\quad\quad\quad v \\
\quad\quad\quad\quad (\mathit{Param\_}\varnothing\ w) \\
\quad\quad\quad\quad = \mathit{EvalForm}\ (\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)\ v\ (\mathit{Param\_}\varnothing\ w)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{Sf\_MsExtend2\_lemma1}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge y \in V \wedge v \in V \wedge w \in V \\
\quad\quad\quad \Rightarrow \mathit{Sf}\ V\ (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y)\ v\ w \\
\quad\quad\quad\quad = \mathit{Sf}\ V\ \$\in_v\ v\ w
\end{array}
$$

The following lemmas tell us some obvious facts about the value assigned to the empty set.

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\varnothing_{\boldsymbol{g}}\textbf{\textit{\_Lub\_lemma1}} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge \mathit{CompExt}\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow \mathit{MsExtend2}\ (V,\ r)\ x\ y\ z\ \varnothing_g \\
\quad\quad\quad\quad = \mathit{Lub}\ \$\leqslant_{t4}\ \{\mathit{MsExtend2}\ (V,\ r)\ x\ y\ z\ x;\ \mathit{MsExtend2}\ (V,\ r)\ x\ y\ z\ y\}
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\varnothing_{\boldsymbol{g}}\textbf{\textit{\_Lub\_lemma2}} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge \mathit{CompExt}\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow \mathit{MsExtend2}\ (V,\ r)\ x\ y\ \varnothing_g\ z \\
\quad\quad\quad\quad = \mathit{Lub}\ \$\leqslant_{t4}\ \{\mathit{MsExtend2}\ (V,\ r)\ x\ y\ x\ z;\ \mathit{MsExtend2}\ (V,\ r)\ x\ y\ y\ z\}
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma1} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge CompEss\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ x\ z \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma1b} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge Compatible\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \cup \{\varnothing_g\} \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ x\ z \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma1} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge CompEss\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ y\ z \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma1b} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge Compatible\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \cup \{\varnothing_g\} \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ y\ z \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma2} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge CompExt\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ z\ x \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma2} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \wedge CompExt\ V\ r\ x\ y \wedge x \in V \wedge y \in V \wedge z \in V \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ z\ y \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma2b} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq Syntax \\
\quad\quad\quad\quad \wedge\ CompEss\ V\ r\ x\ y \\
\quad\quad\quad\quad \wedge\ CompExt\ V\ r\ x\ y \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ z \in V \cup \{\varnothing_g\} \\
\quad\quad\quad \Rightarrow MsExtend2\ (V,\ r)\ x\ y\ z\ x \leqslant_{t4} MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_y\_}}\leqslant_{t4}\textbf{\textit{\_}}\varnothing_g\textbf{\textit{\_lemma2b}} = \\
\quad \vdash \forall\ (V,\ r)\ x\ y\ z \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ r\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ r\ x\ y \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ z \in V \cup \{\varnothing_g\} \\
\quad\quad\quad\quad \Rightarrow \mathit{MsExtend2}\ (V,\ r)\ x\ y\ z\ y \leqslant_{t4} \mathit{MsExtend2}\ (V,\ r)\ x\ y\ z\ \varnothing_g
\end{array}
$$

The extended structure is not anywhere overdefined. Lemma4 subsumes all the others but was not true for my first definition of the extension, which is why I proved the earlier more restricted results.

$$
\begin{array}{l}
\textbf{\textit{MsExtend\_}}\neg\textbf{\textit{OverDefined\_lemma}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow \neg\ \mathit{OverDefined}\ V\ (\mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\neg\textbf{\textit{OverDefined\_lemma}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow \neg\ \mathit{OverDefined}\ V\ (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend\_}}\neg\textbf{\textit{OverDefined\_lemma2}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow (\forall\ v\bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y\ v\ \varnothing_g = fT)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\neg\textbf{\textit{OverDefined\_lemma2}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow (\forall\ v\bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y\ v\ \varnothing_g = fT)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\neg\textbf{\textit{OverDefined\_lemma2}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow (\forall\ v\bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y\ v\ \varnothing_g = fT)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend\_}}\neg\textbf{\textit{OverDefined\_lemma3}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow (\forall\ v\bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y\ \varnothing_g\ v = fT)
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{MsExtend2\_}}\neg\textbf{\textit{OverDefined\_lemma3}} = \\
\quad \vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \Rightarrow (\forall\ v\bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y\ \varnothing_g\ v = fT)
\end{array}
$$

$$\mathbf{MsExtend\_\neg OverDefined\_lemma4} =$$
$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq \mathit{SetReps}$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v$$
$$\wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y$$
$$\Rightarrow\ \neg\ \mathit{OverDefined}\ (V \cup \{\varnothing_g\})\ (\mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y)$$

$$\mathbf{MsExtend2\_\neg OverDefined\_lemma4} =$$
$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq \mathit{Syntax}$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v$$
$$\wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y$$
$$\Rightarrow\ \neg\ \mathit{OverDefined}\ (V \cup \{\varnothing_g\})\ (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y)$$

3.3.3 Strictness-like Properties

A function is strict if is maps bottom to bottom.

A weaker notion we call "weak strictness" and name *WkStrict* which is the property of preserving "UnderDefined"ness. This is formulated for semantic functors.

HOL Constant

$$\mathbf{WkStrict} : \mathit{GS\ SET} \to (\mathit{GS\ SET} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR}) \to \mathit{BOOL}$$

$$\forall V\ f\bullet\ \mathit{WkStrict}\ V\ f \Leftrightarrow \forall r\bullet\ \mathit{UnderDefined}\ V\ r \Rightarrow \mathit{UnderDefined}\ V\ (f\ V\ r)$$

The dual concept is:

HOL Constant

$$\mathbf{CoWkStrict} : \mathit{GS\ SET} \to (\mathit{GS\ SET} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR}) \to \mathit{BOOL}$$

$$\forall V\ f\bullet\ \mathit{CoWkStrict}\ V\ f \Leftrightarrow \forall r\bullet\ \mathit{OverDefined}\ V\ r \Rightarrow \mathit{OverDefined}\ V\ (f\ V\ r)$$

Their "converses" are:

HOL Constant

$$\mathbf{ConWkStrict} : \mathit{GS\ SET} \to (\mathit{GS\ SET} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR} \to (\mathit{GS},\ \mathit{FTV})\mathit{BR}) \to \mathit{BOOL}$$

$$\forall V\ f\bullet\ \mathit{ConWkStrict}\ V\ f \Leftrightarrow \forall r\bullet\ \mathit{UnderDefined}\ V\ (f\ V\ r) \Rightarrow \mathit{UnderDefined}\ V\ r$$

ConWkStrictSf_lemma =

$\vdash \forall\ V \bullet\ V \subseteq SetReps \Rightarrow ConWkStrict\ V\ Sf$

HOL Constant

ConCoWkStrict : $GS\ SET \to (GS\ SET \to (GS,\ FTV)BR \to (GS,\ FTV)BR) \to BOOL$

$\forall V\ f \bullet\ ConCoWkStrict\ V\ f \Leftrightarrow \forall r \bullet\ OverDefined\ V\ (f\ V\ r) \Rightarrow OverDefined\ V\ r$

ConCoWkStrictSf_lemma =

$\vdash \forall\ V \bullet\ V \subseteq Syntax \Rightarrow ConCoWkStrict\ V\ Sf$

ODE_SF_lemma =

$\vdash \forall\ V\ \$\in_v$

- $V \subseteq SetReps$
  $\wedge\ OverDefinedEss\ (V \cup \{\varnothing_g\})\ V\ (Sf\ (V \cup \{\varnothing_g\})\ \$\in_v)$
  $\Rightarrow OverDefined\ (V \cup \{\varnothing_g\})\ \$\in_v$

ODE_SF_lemma2 =

$\vdash \forall\ V\ \$\in_v$

- $V \subseteq SetReps \wedge OverDefinedEss\ (V \cup \{\varnothing_g\})\ V\ (Sf\ V\ \$\in_v)$
  $\Rightarrow OverDefined\ (V \cup \{\varnothing_g\})\ \$\in_v$

ODE_SF_MsExtend2_lemma =

$\vdash \forall\ V\ r\ x\ y$

- $V \subseteq SetReps$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ Compatible\ V\ r\ x\ y$
  $\wedge\ OverDefinedEss$
  $(V \cup \{\varnothing_g\})$
  V
  $(Sf\ (V \cup \{\varnothing_g\})\ (MsExtend2\ (V,\ r)\ x\ y))$
  $\Rightarrow OverDefined\ V\ r$

ODE_SF_MsExtend2_lemma2 =

$\vdash \forall\ V\ r\ x\ y$

- $V \subseteq SetReps$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ Compatible\ V\ r\ x\ y$
  $\wedge\ OverDefinedEss$
  $(V \cup \{\varnothing_g\})$
  V
  $(Sf\ V\ (MsExtend2\ (V,\ r)\ x\ y))$
  $\Rightarrow OverDefined\ V\ r$

$$
\begin{array}{l}
\textbf{\textit{ExsSRO\_MsExtend2\_lemma1}} = \\
\quad \vdash \forall\ V\ r\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge y \in V \wedge \mathit{Compatible}\ V\ r\ x\ y \\
\quad\quad\quad \Rightarrow \mathit{ExsSRO}\ (V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)\ x\ \varnothing_g \\
\quad\quad\quad\quad \wedge\ \mathit{ExsSRO}\ (V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)\ y\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{ExsSRO\_MsExtend2\_lemma2}} = \\
\quad \vdash \forall\ V\ r\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge y \in V \wedge \mathit{Compatible}\ V\ r\ x\ y \\
\quad\quad\quad \Rightarrow \mathit{ExsSRO}\ (V \cup \{\varnothing_g\},\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)\ x\ \varnothing_g \\
\quad\quad\quad\quad \wedge\ \mathit{ExsSRO}\ (V \cup \{\varnothing_g\},\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)\ y\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{Compatibility\_lemma1}} = \\
\quad \vdash \forall\ V\ \$\in_v\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge y \in V \wedge \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad \Rightarrow \mathit{EssSRO}\ (V,\ \mathit{Sf}\ V\ (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y))\ x\ \varnothing_g \\
\quad\quad\quad\quad \wedge\ \mathit{EssSRO}\ (V,\ \mathit{Sf}\ V\ (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y))\ y\ \varnothing_g
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{Compatibility\_lemma2}} = \\
\quad \vdash \forall\ V\ \$\in_v\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \Rightarrow \mathit{CompEss}\ V\ (\mathit{Sf}\ V\ \$\in_v)\ x\ y
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{Compatibility\_lemma3}} = \\
\quad \vdash \forall\ V\ \$\in_v\ x\ y \\
\quad\quad \bullet\ V \subseteq \mathit{SetReps} \\
\quad\quad\quad\quad \wedge\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\quad\quad\quad\quad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\quad\quad\quad\quad \Rightarrow \mathit{let}\ r = (\mathit{Sf}\ V\ \$\in_v)\ \mathit{in}\ \{r\ x\ y;\ r\ y\ x;\ r\ x\ x;\ r\ y\ y\} \in \mathit{CompFTV}
\end{array}
$$

$$
\begin{array}{l}
\textbf{\textit{CompExtSem\_Sf\_thm}} = \\
\quad\quad \vdash \forall\ V\bullet\ V \subseteq \mathit{SetReps} \Rightarrow \mathit{CompExtSem}\ V\ \mathit{Sf}
\end{array}
$$

3.4 Properties of SfChains

Properties of least fixed points of the semantic functor are in part obtained by induction over the relevant FChain (see [1] for the theory of $FChains$). Since inductive proofs yeild properties shared by all the members of the FChain, and its best to prove them of the FChain so that their full force can be used in subsequent inductive proofs.

First abbreviated notation for the FChains generated by the semantic functor.

HOL Constant

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

$$\forall V \bullet\ SfChain\ V\ =\ FChainU\ (Sf\ V)\ \$\leqslant_\epsilon$$

For inductive reasoning about $SfChain$s the general induction theorem for fchains is specialised to the chains generated by the semantic functor.

$$\begin{array}{l}
\boldsymbol{sfchain_induction_thm} = \\
\quad \vdash\ \forall\ V\ p \\
\quad\quad \bullet\ V \subseteq SetReps \\
\quad\quad\quad \wedge\ (\forall\ x \bullet\ x \in SfChain\ V\ \wedge\ (\forall\ y \bullet\ y \in SfChain\ V\ \wedge\ y \leqslant_\epsilon x \Rightarrow p\ y) \\
\quad\quad\quad\quad \Rightarrow p\ (Sf\ V\ x)) \\
\quad\quad\quad \wedge\ (\forall\ x \bullet\ x \in SfChain\ V \\
\quad\quad\quad\quad \wedge\ x = Lub\ \$\leqslant_\epsilon\ \{y | y \in SfChain\ V\ \wedge\ y \leqslant_\epsilon x\ \wedge\ \neg\ x \leqslant_\epsilon y\} \\
\quad\quad\quad\quad \wedge\ (\forall\ y \bullet\ y \in SfChain\ V\ \wedge\ y \leqslant_\epsilon x\ \wedge\ \neg\ x \leqslant_\epsilon y \Rightarrow p\ y) \\
\quad\quad\quad\quad \Rightarrow p\ x) \\
\quad\quad\quad \Rightarrow (\forall\ x \bullet\ x \in SfChain\ V \Rightarrow p\ x)
\end{array}$$

It will be convenient to have a tactic to facilitate this kind of induction.

The following tactic does not expect an argument, but expects the conclusion of the goal to be an implication of which the antecedent asserts that the induction variable is a member of the relevant $SfChain$.

SML

```
fun sfchain_induction_tac (a,c) =
      let val (l1, l2) = dest_⇒ c
      in let val (_, [v, _]) = strip_app l1
      in let val l3 = mk_∀ (v, mk_⇒ (l1, mk_app (mk_λ (v, l2), v)))
      in
            LEMMA_T l3 (rewrite_thm_tac o (rewrite_rule []))
            THEN bc_tac [sfchain_induction_thm]
            THEN REPEAT strip_tac THEN_TRY rewrite_tac[]
      end end end (a,c);
```

This kind of inductive proof requires reasoning about the least upper bounds of subsets of $SfChain$s for which the following lemmas will be helpful.

Some general properties of $FChains$ are specialised to $SfChain$s for convenience of application.

$\mathbf{sfchain\_mono\_thm} =$
$\vdash \forall\ V\ x \bullet\ x \in SfChain\ V \Rightarrow x \leqslant_{\in} Sf\ V\ x)$

$\mathbf{sfchain\_linear\_lemma} =$
$\vdash \forall\ V \bullet\ LinearOrder\ (SfChain\ V, \$\leqslant_{\in})$

$\mathbf{sfsubchain\_linear\_lemma} =$
$\vdash \forall\ V\ X \bullet\ X \subseteq SfChain\ V \Rightarrow LinearOrder\ (X, \$\leqslant_{\in})$

The following properties of $SfChain$s are needed to prove that least fixed points are "pre-extensional".

The first property we consider is that of non-overdefinedness.

$\mathbf{sfchain\_\neg overdefined\_lemma} =$
$\vdash \forall\ V \bullet\ V \subseteq SetReps \Rightarrow (\forall\ x \bullet\ x \in SfChain\ V \Rightarrow \neg\ OverDefined\ V\ x)$

Now we derive the properties of $SfChain$s which follow from the $CompExtSem$ property of the semantic functor.

SML
set_flag ("*subgoal_package_quiet*", *false*);

4 The Theory ifos

4.1 Parents

membership misc2

4.2 Children

sfp

4.3 Constants

$\boldsymbol{MkAf}$	$(GS,\ GS \times GS)\ VA$
$\boldsymbol{IsAf}$	$(BOOL,\ GS)\ VA$
$\boldsymbol{AfSet}$	$(GS,\ GS)\ VA$
$\boldsymbol{AfMem}$	$(GS,\ GS)\ VA$
$\boldsymbol{MkCf}$	$(GS,\ GS \times GS)\ VA$
$\boldsymbol{IsCf}$	$(BOOL,\ GS)\ VA$
$\boldsymbol{CfVars}$	$(GS,\ GS)\ VA$
$\boldsymbol{CfForms}$	$(GS,\ GS)\ VA$
$\boldsymbol{MkNot}$	$(GS,\ GS)\ VA$
$\boldsymbol{RepClosed}$	$(BOOL,\ GS\ \mathbb{P})\ VA$
$\boldsymbol{RepOpen}$	$(BOOL,\ GS\ \mathbb{P})\ VA$
$\boldsymbol{Syntax}$	$GS\ \mathbb{P}$
$\boldsymbol{CoSyntax}$	$GS\ \mathbb{P}$
$\boldsymbol{ScPrec}$	$((BOOL,\ GS)\ VA,\ GS)\ VA$
$\boldsymbol{ScPrec2}$	$((BOOL,\ GS)\ VA,\ GS)\ VA$
$\boldsymbol{FreeVars2}$	$(GS\ \mathbb{P},\ GS)\ VA$
$\boldsymbol{FreeVarsFunct}$	$((GS\ \mathbb{P},\ GS)\ VA,\ (GS\ \mathbb{P},\ GS)\ VA)\ VA$
$\boldsymbol{FreeVars}$	$(GS\ \mathbb{P},\ GS)\ VA$
$\boldsymbol{SetReps}$	$GS\ \mathbb{P}$
$\boldsymbol{EvalAf}$	$((((''a,\ ''t)\ RV,\ (''a,\ ''t)\ ST)\ VA,\ GS)\ VA,$ $((BOOL,\ ''t)\ VA,\ ''t)\ VA)\ VA$
$\boldsymbol{EvalCf\_tf3}$	$TTV\ \ CFE$
$\boldsymbol{EvalCf\_bool}$	$BOOL\ CFE$
$\boldsymbol{LiftEvalCf\_bool}$	$BOOL\ \mathbb{P}\ CFE$
$\boldsymbol{BoolSet2FTV}$	$(FTV,\ BOOL\ \mathbb{P})\ VA$
$\boldsymbol{FTV2BoolSet}$	$(BOOL\ \mathbb{P},\ FTV)\ VA$
$\boldsymbol{EvalCf2\_ftv}$	$FTV\ \ CFE$
$\boldsymbol{EvalCf\_ftv}$	$FTV\ \ CFE$
$\boldsymbol{EvalCf}$	$((((''a,\ ''t)\ RV\ \ CFE,\ (''a,\ ''t)\ ST)\ VA,\ GS)\ VA,\ ''t\ \ CFE)\ VA$
$\boldsymbol{EvalFormFunct}$	$((((''a,\ ''t)\ RV,\ GS)\ VA,\ ((''a,\ ''t)\ RV,\ GS)\ VA)\ VA,$ $''t\ \ CFE \times ((BOOL,\ ''t)\ VA,\ ''t)\ VA \times (''a,\ ''t)\ ST)\ VA$
$\boldsymbol{EvalForm}$	$(((''a,\ ''t)\ RV,\ GS)\ VA,$ $''t\ \ CFE \times ((BOOL,\ ''t)\ VA,\ ''t)\ VA \times (''a,\ ''t)\ ST)\ VA$
$\boldsymbol{PmrEq}$	$(((BOOL,\ (('b,\ ''a)\ VA,\ ''a)\ VA)\ VA,$ $(('b,\ ''a)\ VA,\ ''a)\ VA)\ VA,$ $''a\ \mathbb{P})\ VA$

RvO $(((BOOL,\ ('a,\ 'b)\ RV)\ VA,\ ('a,\ 'b)\ RV)\ VA,$
$((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$

RvIsO $(((BOOL,\ (('a,\ 'b)\ RV\ DPO,\ GS)\ VA)\ VA,$
$(('a,\ 'b)\ RV\ DPO,\ GS)\ VA)\ VA,$
$((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$

StO $(((BOOL,\ ('a,\ 'b)\ ST)\ VA,\ ('a,\ 'b)\ ST)\ VA,$
$((BOOL,\ 'b)\ VA,\ 'b)\ VA)\ VA$

$\$\leqslant_{ft3}$ $((BOOL,\ (TTV,\ 'a)\ VA)\ VA,\ (TTV,\ 'a)\ VA)\ VA$

$\$\leqslant_{ft4}$ $((BOOL,\ (FTV,\ 'a)\ VA)\ VA,\ (FTV,\ 'a)\ VA)\ VA$

ExtSRO $(((BOOL,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

EssSRO $(((BOOL,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

ExsSRO $(((BOOL,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

ExsVaO $(((BOOL,\ (GS\ OPT,\ 'a)\ VA)\ VA,\ (GS\ OPT,\ 'a)\ VA)\ VA,$
$(GS\ OPT,\ 'a)\ VA\ \mathbb{P} \times (GS,\ FTV)\ ST)\ VA$

V2IxSet $((GS\ OPT,\ 'a)\ VA\ \mathbb{P},\ GS\ \mathbb{P})\ VA$

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

Extension $(((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA$

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

Essence $(((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA$

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

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

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

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

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

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

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

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

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

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

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

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

4.4 Type Abbreviations

$('a,\ 'b)$ ***RV*** $('a,\ 'b)\ RV$

$('a,\ 'b)$ ***ST*** $('a,\ 'b)\ ST$

$'t$ ***CFE*** $'t\ CFE$

4.5 Fixity

Right Infix 300:

$\in_{\boldsymbol{v}} \quad \leqslant_{\boldsymbol{ft3}} \leqslant_{\boldsymbol{ft4}}$

4.6 Definitions

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

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

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

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

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

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

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

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

FreeVars2
$\vdash (\forall\ x\ y \bullet\ FreeVars2\ (MkAf\ (x, y)) = \{x; y\})$
$\wedge\ (\forall\ vars\ forms$
$\bullet\ FreeVars2\ (MkCf\ (vars, forms))$
$= \bigcup (FunImage_g\ FreeVars2\ forms) \setminus X_g\ vars)$

FreeVarsFunct
$\vdash FreeVarsFunct$
$= (\lambda\ fv\ f$
$\bullet\ if\ f \in Syntax$
$then$
$if\ IsAf\ f$
$then\ \{AfMem\ f;\ AfSet\ f\}$
$else$
$\bigcup (FunImage_g\ fv\ (CfForms\ f))$
$\setminus X_g\ (CfVars\ f)$
$else\ \epsilon\ x \bullet\ T)$

FreeVars $\vdash FreeVars = fix\ FreeVarsFunct$

SetReps $\vdash SetReps = \{x | x \in Syntax \wedge FreeVars\ x = \{\varnothing_g\}\}$

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

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

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

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

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

FTV2BoolSet $\vdash \forall\ ftv$
- $FTV2BoolSet\ ftv$
  - $= \{x$
    $|(x \Leftrightarrow T) \wedge fTrue \leqslant_{t4} ftv$
    $\vee\ (x \Leftrightarrow F) \wedge fFalse \leqslant_{t4} ftv\}$

Evalcf2_ftv $\vdash \forall\ results$
- $Evalcf2\_ftv\ results$
  - $= BoolSet2FTV$
    $(LiftEvalCf\_bool$
    $(FunImage\ FTV2BoolSet\ results))$

EvalCf_ftv $\vdash \forall\ results$
- $EvalCf\_ftv\ results$
  - $= Lub$
    $\$\leqslant_{t4}$
    $\{t$
    $|(fFalse \in results \vee fT \in results)$
    $\wedge\ t = fTrue$
    $\vee\ (\neg\ fFalse \in results \wedge \neg\ fB \in results)$

$$\wedge\ t = fFalse\}$$

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

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

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

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

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

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

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

$\leqslant_{ft3}$ $\vdash \$\leqslant_{ft3} = Pw\ \$\leqslant_{t3}$

$\leqslant_{ft4}$ $\vdash \$\leqslant_{ft4} = Pw\ \$\leqslant_{t4}$

ExtSRO $\vdash \forall\ V\ r$
$\bullet\ ExtSRO\ (V, r)$
$= (\lambda\ x\ y\bullet\ \forall\ z\bullet\ z \in V \Rightarrow r\ z\ x \leqslant_{t4} r\ z\ y)$

EssSRO $\vdash \forall\ V\ r$
$\bullet\ EssSRO\ (V, r)$
$= (\lambda\ x\ y\bullet\ \forall\ z\bullet\ z \in V \Rightarrow r\ x\ z \leqslant_{t4} r\ y\ z)$

ExsSRO $\vdash \forall\ s\bullet\ ExsSRO\ s = ConjOrder\ (ExtSRO\ s)\ (EssSRO\ s)$

ExsVaO $\vdash \forall\ d\ s\bullet\ ExsVaO\ (d, s) = IxO2\ (d, ExsSRO\ s)$

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

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

Extension $\vdash Extension = CombC$

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

Essence $\vdash Essence = CombI$

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

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

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

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

4.7 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
$\vdash (\forall\ s2\ m \bullet\ MkAf\ (s2,\ m) \in Syntax)$
$\wedge\ (\forall\ vars\ fs$
$\bullet\ (\forall\ x \bullet\ x \in X_g\ fs \Rightarrow x \in Syntax)$
$\Rightarrow MkCf\ (vars,\ fs) \in Syntax)$

repclosed_syntax_thm2

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

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

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

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

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

syn_proj_clauses

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

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_con_neq_clauses

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

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

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

sc_recursion_lemma

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

syn_induction_thm

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

freevarsfunct_respect_thm

$$\vdash FreeVarsFunct\ respects\ ScPrec$$

freevarsfunct_fixp_lemma

$$\vdash FreeVars = FreeVarsFunct\ FreeVars$$

freevarsfunct_thm

$$\vdash FreeVars$$
$$= (\lambda\ f$$
$$\bullet\ if\ f \in Syntax$$
$$then$$
$$if\ IsAf\ f$$
$$then\ \{AfMem\ f;\ AfSet\ f\}$$
$$else$$
$$\bigcup\ (FunImage_g\ FreeVars\ (CfForms\ f))$$
$$\setminus\ X_g\ (CfVars\ f)$$

$$else\ \epsilon\ x\bullet\ T)$$

freevarsfunct_thm2

$$\vdash\ \forall\ f$$
$$\bullet\ FreeVars\ f$$
$$= (if\ f \in Syntax$$
$$then$$
$$if\ IsAf\ f$$
$$then\ \{AfMem\ f;\ AfSet\ f\}$$
$$else$$
$$\bigcup\ (FunImage_g\ FreeVars\ (CfForms\ f))$$
$$\setminus\ X_g\ (CfVars\ f)$$
$$else\ \epsilon\ x\bullet\ T)$$

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

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

evalformfunct_fixp_lemma

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

$(EvalForm\ (cfe,\ \leqslant_t,\ st))$

evalformfunct_thm

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

evalformfunct_thm2

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

EvalForm_fT_lemma

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

EvalForm_fT_lemma2

$\vdash\ \forall\ y$
- $y \in Syntax$
  $\Rightarrow (\forall\ va$
  - $FreeVars\ y \subseteq IxDom\ va$
      $\wedge\ IxRan\ va \subseteq V\ \cup\ \{\varnothing_g\}$

$$\wedge\ EvalForm$$
$$(EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ \$\in_v)$$
$$y$$
$$va$$
$$= fT$$
$$\Rightarrow (\exists\ x\ y$$
$$\bullet\ x \in V \cup \{\varnothing_g\}$$
$$\wedge\ y \in V \cup \{\varnothing_g\}$$
$$\wedge\ x \in_v y = fT))$$

PmrEq_EvalForm_lemma

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

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_$\leqslant_{ft3}$_thm

$$\vdash CcRpoU\ \$\leqslant_{ft3}$$

$\leqslant_{ft4}$_crpou_thm

$$\vdash CRpoU\ \$\leqslant_{ft4}$$

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

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

evalform_increasing_thm

$$\vdash \forall\ c\ r\ s\ g$$

• $CRpoU\ r \wedge Increasing\ (SetO\ r)\ r\ c$
$\Rightarrow Increasing$
$(Pw\ (Pw\ r))$
$(RvO\ r)$
$(\lambda\ ris\bullet\ EvalForm\ (c,\ r,\ s,\ ris)\ g)$

evalform_increasing_thm2

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

evalform_increasing_thm3

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

evalform_increasing_thm4

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

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

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

samess_sym_lemma

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

OverDefinedL_$\leqslant_{t4}$_lemma

$\vdash \forall\ V\ xe1\ xe2$
• $\neg\ OverDefinedL\ V\ xe1 \wedge PwS\ V\ \$\leqslant_{t4}\ xe2\ xe1$
$\Rightarrow \neg\ OverDefinedL\ V\ xe2$

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

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

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

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

compext_sym_lemma

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

cocompext_sym_lemma

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

compess_sym_lemma

$$\vdash \forall\ V\ r$$
$$\bullet\ \neg\ \mathit{OverDefined}\ V\ r$$
$$\Rightarrow (\forall\ x\ y$$
$$\bullet\ x \in V \wedge y \in V$$
$$\Rightarrow (\mathit{CompEss}\ V\ r\ x\ y \Leftrightarrow \mathit{CompEss}\ V\ r\ y\ x))$$

cocompess_sym_lemma

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

Compatible_lemma1

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

5 The Theory sfp

5.1 Parents

ifos

5.2 Constants

Param_∅	$((GS\ OPT,\ GS)\ VA,\ GS)\ VA$
Sf	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
SFFixp	$((BOOL,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
TotalOver	$((BOOL,\ ((FTV,\ 'a)\ VA,\ 'a)\ VA)\ VA,\ 'a\ \mathbb{P})\ VA$
BoolRel	$(((BOOL,\ 'a)\ VA,\ 'a)\ VA,\ ((FTV,\ 'a)\ VA,\ 'a)\ VA)\ VA$
BEqRel	$((((BOOL,\ GS)\ VA,\ GS)\ VA,\ ((BOOL,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
BEqqRel	$((((BOOL,\ GS)\ VA,\ GS)\ VA,\ ((BOOL,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
BoolEqRel	$((((BOOL,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
BoolEqqRel	$((((BOOL,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
PreExtensional	$((BOOL,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
PreExtensional2	$((BOOL,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
PreExtensional3	$((BOOL,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
TotExtensional	$((BOOL,\ ((BOOL,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
TotExtensional2	$((BOOL,\ ((BOOL,\ GS)\ VA,\ GS)\ VA)\ VA,\ GS\ \mathbb{P})\ VA$
MSfromSFF	$((GS\ \mathbb{P},\ BOOL)\ ST,\ (GS,\ FTV)\ ST)\ VA$
MSfromSFF2	$((GS\ \mathbb{P},\ BOOL)\ ST,\ (GS,\ FTV)\ ST)\ VA$
ExtSem	$((BOOL,$ $((((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
CompExtSem	$((BOOL,$ $((((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
CompExtSem2	$((BOOL,$ $((((FTV,\ GS)\ VA,\ GS)\ VA,\ ((FTV,\ GS)\ VA,\ GS)\ VA)\ VA,$ $GS\ \mathbb{P})\ VA)\ VA,$ $GS\ \mathbb{P})\ VA$
ExtUb	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$
ExtUb2	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$
ExtLub	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$
ExtLub2	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$
EssUb	$((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

EssLub $((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

EssLub2 $((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

MsExtend $(((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

MsExtend2 $(((((FTV,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ GS)\ VA,\ (GS,\ FTV)\ ST)\ VA$

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

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

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

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

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

5.3 Definitions

Param_∅ $\vdash \forall\ p \bullet\ Param\_\varnothing\ p = (\lambda\ x \bullet\ if\ x = \varnothing_g\ then\ Value\ p\ else\ Undefined)$

Sf $\vdash \forall\ d\ \$\in_v \bullet\ Sf\ d\ \$\in_v = (\lambda\ m\ s \bullet\ EvalForm\ (EvalCf\_ftv,\ \$\leqslant_{t4},\ d,\ \$\in_v)\ s\ (Param\_\varnothing\ m))$

SFFixp $\vdash \forall\ d\ r \bullet\ SFFixp\ d\ r \Leftrightarrow Sf\ d\ r = r$

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

BoolRel $\vdash \forall\ r \bullet\ BoolRel\ r = (\lambda\ x\ y \bullet\ fTrue \leqslant_{t4} r\ x\ y)$

BEqRel $\vdash \forall\ V\ r \bullet\ BEqRel\ V\ r = (\lambda\ x\ y \bullet\ \forall\ z \bullet\ z \in V \Rightarrow (r\ z\ x \Leftrightarrow r\ z\ y))$

BEqqRel $\vdash \forall\ V\ r \bullet\ BEqqRel\ V\ r = (\lambda\ x\ y \bullet\ \forall\ z \bullet\ z \in V \Rightarrow (r\ z\ x \Leftrightarrow r\ z\ y) \wedge (r\ x\ z \Leftrightarrow r\ y\ z))$

BoolEqRel $\vdash \forall\ V\ r \bullet\ BoolEqRel\ V\ r = BEqRel\ V\ (BoolRel\ r)$

BoolEqqRel $\vdash \forall\ V\ r \bullet\ BoolEqqRel\ V\ r = BEqqRel\ V\ (BoolRel\ r)$

PreExtensional

$$
\begin{aligned}
&\vdash \forall\ d\ r \\
&\quad \bullet\ PreExtensional\ d\ r \\
&\qquad \Leftrightarrow \neg\ OverDefined\ d\ r \\
&\qquad\quad \wedge\ (\forall\ x\ y \\
&\qquad\quad \bullet\ x \in d \wedge y \in d \\
&\qquad\qquad \Rightarrow (\exists\ z \\
&\qquad\qquad \bullet\ z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ x\ z;\ r\ y\ z\}\ fT) \\
&\qquad\qquad \Rightarrow (\exists\ z \\
&\qquad\qquad \bullet\ z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ z\ x;\ r\ z\ y\}\ fT))
\end{aligned}
$$

PreExtensional2

$$
\begin{aligned}
&\vdash \forall\ d\ r \\
&\quad \bullet\ PreExtensional2\ d\ r \\
&\qquad \Leftrightarrow \neg\ OverDefined\ d\ r \\
&\qquad\quad \wedge\ (\forall\ x\ y \\
&\qquad\quad \bullet\ x \in d \wedge y \in d \\
&\qquad\qquad \Rightarrow (\exists\ z \\
&\qquad\qquad \bullet\ z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ x\ z;\ r\ y\ z\}\ fT) \\
&\qquad\qquad \Rightarrow (\exists\ z \\
&\qquad\qquad\quad \bullet\ z \in d \wedge IsLub\ \$\leqslant_{t4}\ \{r\ z\ x;\ r\ z\ y\}\ fT) \\
&\qquad\qquad\quad \vee\ Lub\ \$\leqslant_{t4}\ \{r\ x\ x;\ r\ y\ y\} = fT \\
&\qquad\qquad\quad \vee\ Lub\ \$\leqslant_{t4}\ \{r\ x\ y;\ r\ y\ x\} = fT)
\end{aligned}
$$

PreExtensional3

$$
\begin{aligned}
&\vdash \forall\ d\ r \\
&\quad \bullet\ PreExtensional3\ d\ r \\
&\qquad \Leftrightarrow \neg\ OverDefined\ d\ r \\
&\qquad\quad \wedge\ (\forall\ x\ y \\
&\qquad\quad \bullet\ x \in d \wedge y \in d \\
&\qquad\qquad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ \{r\ x\ z;\ r\ y\ z\} \in CompFTV) \\
&\qquad\qquad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ \{r\ z\ x;\ r\ z\ y\} \in CompFTV) \\
&\qquad\qquad\quad \vee\ \neg\ \{r\ x\ x;\ r\ y\ y\} \in CompFTV \\
&\qquad\qquad\quad \vee\ \neg\ \{r\ x\ y;\ r\ y\ x\} \in CompFTV)
\end{aligned}
$$

TotExtensional

$$
\begin{aligned}
&\vdash \forall\ d\ r \\
&\quad \bullet\ TotExtensional\ d\ r \\
&\qquad \Leftrightarrow (\forall\ x\ y \\
&\qquad \bullet\ x \in d \wedge y \in d \\
&\qquad\quad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ (r\ x\ z \Leftrightarrow r\ y\ z)) \\
&\qquad\quad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ (r\ z\ x \Leftrightarrow r\ z\ y)))
\end{aligned}
$$

TotExtensional2

$$
\begin{aligned}
&\vdash \forall\ d\ r \\
&\quad \bullet\ TotExtensional2\ d\ r \\
&\qquad \Leftrightarrow (\forall\ x\ y \\
&\qquad \bullet\ x \in d \wedge y \in d \\
&\qquad\quad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ (r\ x\ z \Leftrightarrow r\ y\ z)) \\
&\qquad\quad \Rightarrow (\exists\ z \bullet\ z \in d \wedge \neg\ (r\ z\ x \Leftrightarrow r\ z\ y)) \\
&\qquad\qquad \vee\ \neg\ (r\ x\ x \Leftrightarrow r\ y\ y) \\
&\qquad\qquad \vee\ \neg\ (r\ x\ y \Leftrightarrow r\ y\ x))
\end{aligned}
$$

MSfromSFF $\vdash \forall\ V\ r$

$$
\begin{array}{l}
\quad \bullet\ \mathit{MSfromSFF}\ (V,\ r) \\
\quad\quad = (\mathit{let}\ \$\in_v\ =\ \mathit{BoolRel}\ r \\
\quad\quad\ \ \mathit{and}\ =_v\ =\ \mathit{BEqRel}\ V\ (\mathit{BoolRel}\ r) \\
\quad\quad\ \ \mathit{in}\ (V\ /\ =_v, \\
\quad\quad\quad (\lambda\ s\ t \bullet\ \forall\ x\ y \bullet\ x \in s\ \wedge\ y \in t \Rightarrow x \in_v y)))
\end{array}
$$

MSfromSFF2 $\vdash\ \forall\ V\ r$

$$
\begin{array}{l}
\quad \bullet\ \mathit{MSfromSFF2}\ (V,\ r) \\
\quad\quad = (\mathit{let}\ \$\in_v\ =\ \mathit{BoolRel}\ r \\
\quad\quad\ \ \mathit{and}\ =_v\ =\ \mathit{BEqqRel}\ V\ (\mathit{BoolRel}\ r) \\
\quad\quad\ \ \mathit{in}\ (V\ /\ =_v, \\
\quad\quad\quad (\lambda\ s\ t \bullet\ \forall\ x\ y \bullet\ x \in s\ \wedge\ y \in t \Rightarrow x \in_v y)))
\end{array}
$$

ExtSem $\vdash\ \forall\ V\ f$

$$
\begin{array}{l}
\quad \bullet\ \mathit{ExtSem}\ V\ f \\
\quad\quad \Leftrightarrow (\forall\ r\ x\ y \\
\quad\quad \bullet\ x \in V \\
\quad\quad\quad\quad \wedge\ y \in V \\
\quad\quad\quad\quad \wedge\ \mathit{SameExt}\ V\ r\ x\ y \\
\quad\quad\quad\quad \wedge\ \mathit{SameEss}\ V\ r\ x\ y \\
\quad\quad\quad\quad \Rightarrow \mathit{SameEss}\ V\ (f\ V\ r)\ x\ y)
\end{array}
$$

CompExtSem $\vdash\ \forall\ V\ f$

$$
\begin{array}{l}
\quad \bullet\ \mathit{CompExtSem}\ V\ f \\
\quad\quad \Leftrightarrow (\forall\ r \\
\quad\quad \bullet\ \neg\ \mathit{OverDefined}\ V\ r \\
\quad\quad\quad \Rightarrow (\forall\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V\ \wedge\ \mathit{Compatible}\ V\ r\ x\ y \\
\quad\quad\quad\quad \Rightarrow \mathit{CompEss}\ V\ (f\ V\ r)\ x\ y))
\end{array}
$$

CompExtSem2 $\vdash\ \forall\ V\ f$

$$
\begin{array}{l}
\quad \bullet\ \mathit{CompExtSem2}\ V\ f \\
\quad\quad \Leftrightarrow (\forall\ r \\
\quad\quad \bullet\ \neg\ \mathit{OverDefined}\ V\ r \\
\quad\quad\quad \Rightarrow (\forall\ x\ y \\
\quad\quad\quad \bullet\ x \in V\ \wedge\ y \in V\ \wedge\ \mathit{Compatible}\ V\ r\ x\ y \\
\quad\quad\quad\quad \Rightarrow \mathit{CompEss}\ V\ (f\ V\ r)\ x\ y \\
\quad\quad\quad\quad\quad \wedge\ \{r\ x\ x;\ r\ x\ y;\ r\ y\ x;\ r\ y\ y\} \\
\quad\quad\quad\quad\quad\quad \in \mathit{CompFTV}))
\end{array}
$$

ExtUb $\vdash\ \forall\ (V,\ \$\in_v)\ x\ y$

$$
\begin{array}{l}
\quad \bullet\ V \subseteq \mathit{SetReps}\ \wedge\ x \in V\ \wedge\ y \in V\ \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad \Rightarrow (\mathit{let}\ xt\ =\ \mathit{ExtUb}\ (V,\ \$\in_v)\ x\ y \\
\quad\quad\quad \mathit{in}\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad \wedge\ \mathit{IsUb} \\
\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad \{\mathit{Extension}\ \$\in_v\ x;\ \mathit{Extension}\ \$\in_v\ y\} \\
\quad\quad\quad\quad\quad xt)
\end{array}
$$

ExtUb2 $\vdash\ \forall\ (V,\ \$\in_v)\ x\ y$

$$
\begin{array}{l}
\quad \bullet\ V \subseteq \mathit{Syntax}\ \wedge\ x \in V\ \wedge\ y \in V\ \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\quad\quad \Rightarrow (\mathit{let}\ xt\ =\ \mathit{ExtUb2}\ (V,\ \$\in_v)\ x\ y \\
\quad\quad\quad \mathit{in}\ \neg\ \mathit{OverDefinedL}\ V\ xt \\
\quad\quad\quad\quad \wedge\ \mathit{IsUb} \\
\quad\quad\quad\quad\quad (\mathit{PwS}\ V\ \$\leqslant_{t4}) \\
\quad\quad\quad\quad\quad \{\mathit{Extension}\ \$\in_v\ x;\ \mathit{Extension}\ \$\in_v\ y\} \\
\quad\quad\quad\quad\quad xt)
\end{array}
$$

ExtLub $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y$
  $\Rightarrow (let\ xt = ExtLub\ (V,\ \$\in_v)\ x\ y$
  $in\ \neg\ OverDefinedL\ V\ xt$
  $\wedge\ IsLub$
  $(PwS\ V\ \$\leqslant_{t4})$
  $\{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\}$
  $xt)$

ExtLub2 $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge CompExt\ V\ \$\in_v\ x\ y$
  $\Rightarrow (let\ xt = ExtLub2\ (V,\ \$\in_v)\ x\ y$
  $in\ \neg\ OverDefinedL\ V\ xt$
  $\wedge\ IsLub$
  $(PwS\ V\ \$\leqslant_{t4})$
  $\{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\}$
  $xt)$

EssUb $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y$
  $\Rightarrow (let\ es = EssUb\ (V,\ \$\in_v)\ x\ y$
  $in\ \neg\ OverDefinedL\ V\ es$
  $\wedge\ IsUb$
  $(PwS\ V\ \$\leqslant_{t4})$
  $\{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\}$
  $es)$

EssLub $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $V \subseteq SetReps \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y$
  $\Rightarrow (let\ es = EssLub\ (V,\ \$\in_v)\ x\ y$
  $in\ \neg\ OverDefinedL\ V\ es$
  $\wedge\ IsLub$
  $(PwS\ V\ \$\leqslant_{t4})$
  $\{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\}$
  $es)$

EssLub2 $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge CompEss\ V\ \$\in_v\ x\ y$
  $\Rightarrow (let\ es = EssLub2\ (V,\ \$\in_v)\ x\ y$
  $in\ \neg\ OverDefinedL\ V\ es$
  $\wedge\ IsLub$
  $(PwS\ V\ \$\leqslant_{t4})$
  $\{Essence\ \$\in_v\ x;\ Essence\ \$\in_v\ y\}$
  $es)$

MsExtend $\vdash \forall\ (V,\ \$\in_v)\ x\ y$
- $MsExtend\ (V,\ \$\in_v)\ x\ y$
  $= (let\ xt = ExtUb\ (V,\ \$\in_v)\ x\ y$
  $and\ es = EssUb\ (V,\ \$\in_v)\ x\ y$
  $in\ \lambda\ x'\ y'$
  - $if\ x' = \varnothing_g$
    $then$
    $if\ y' = \varnothing_g$
    $then$
    Lub
    $\$\leqslant_{t4}$

$$\{x \in_v x;\ y \in_v x;\ x \in_v y;\ y \in_v y\}$$
$$\textit{else es } y')$$
$$\textit{else if } y' = \varnothing_g$$
$$\textit{then xt } x'$$
$$\textit{else } x' \in_v y')$$

MsExtend2 $\vdash \forall\ (V, \$\in_v)\ x\ y$
$$\bullet\ \textit{MsExtend2}\ (V, \$\in_v)\ x\ y$$
$$= (\textit{let } xt = \textit{ExtLub2}\ (V, \$\in_v)\ x\ y$$
$$\textit{and } es = \textit{EssLub2}\ (V, \$\in_v)\ x\ y$$
$$\textit{in } \lambda\ x'\ y'$$
$$\bullet\ \textit{if } x' = \varnothing_g$$
$$\textit{then}$$
$$\textit{if } y' = \varnothing_g$$
$$\textit{then}$$
$$\textit{Lub}$$
$$\$\leqslant_{t4}$$
$$\{x \in_v x;\ y \in_v x;\ x \in_v y;\ y \in_v y\}$$
$$\textit{else es } y'$$
$$\textit{else if } y' = \varnothing_g$$
$$\textit{then xt } x'$$
$$\textit{else } x' \in_v y')$$

WkStrict $\vdash \forall\ V\ f$
$$\bullet\ \textit{WkStrict}\ V\ f$$
$$\Leftrightarrow (\forall\ r$$
$$\bullet\ \textit{UnderDefined}\ V\ r \Rightarrow \textit{UnderDefined}\ V\ (f\ V\ r))$$

CoWkStrict $\vdash \forall\ V\ f$
$$\bullet\ \textit{CoWkStrict}\ V\ f$$
$$\Leftrightarrow (\forall\ r\bullet\ \textit{OverDefined}\ V\ r \Rightarrow \textit{OverDefined}\ V\ (f\ V\ r))$$

ConWkStrict $\vdash \forall\ V\ f$
$$\bullet\ \textit{ConWkStrict}\ V\ f$$
$$\Leftrightarrow (\forall\ r$$
$$\bullet\ \textit{UnderDefined}\ V\ (f\ V\ r) \Rightarrow \textit{UnderDefined}\ V\ r)$$

ConCoWkStrict
$\vdash \forall\ V\ f$
$$\bullet\ \textit{ConCoWkStrict}\ V\ f$$
$$\Leftrightarrow (\forall\ r\bullet\ \textit{OverDefined}\ V\ (f\ V\ r) \Rightarrow \textit{OverDefined}\ V\ r)$$

SfChain $\vdash \forall\ V\bullet\ \textit{SfChain}\ V = \textit{FChainU}\ (\textit{Sf}\ V)\ \$\leqslant_\in$

5.4 Theorems

ixdom_param_∅_lemma
$\vdash \forall\ x\bullet\ \textit{IxDom}\ (\textit{Param\_}\varnothing\ x) = \{\varnothing_g\}$

ixran_param_∅_lemma
$\vdash \forall\ x\bullet\ \textit{IxRan}\ (\textit{Param\_}\varnothing\ x) = \{x\}$

ixval_param_∅_lemma
$\vdash \forall\ x\bullet\ \textit{IxVal}\ (\textit{Param\_}\varnothing\ x)\ \varnothing_g = x$

valueof_param_∅_lemma
$\vdash \forall\ x\bullet\ \textit{ValueOf}\ (\textit{Param\_}\varnothing\ x\ \varnothing_g) = x$

param_∅_undefined_lemma
$\vdash \forall\ x\ y\bullet\ \textit{Param\_}\varnothing\ x\ y = \textit{Undefined} \Leftrightarrow \neg\ y = \varnothing_g$

Param_∅_Increasing_lemma

$\vdash \forall\ V\ r$
- *Increasing*
  $(V \lhd_o ExsSRO\ (V,\ r))$
  $(ExsVaO\ (V2IxSet\ V,\ V,\ r))$
  $Param\_\varnothing$

Param_∅_Increasing_lemma2

$\vdash \forall\ V\ W\ r$
- *Increasing*
  $(W \lhd_o ExsSRO\ (V,\ r))$
  $(ExsVaO\ (V2IxSet\ W,\ V,\ r))$
  $Param\_\varnothing$

sf_increasing_thm

$\vdash \forall\ d\bullet\ Increasing\ \$\leqslant_\epsilon\ \$\leqslant_\epsilon\ (Sf\ d)$

essence_exstentional_lemma

$\vdash \forall\ V\ W\ \$\in_v$
- $V \subseteq W \wedge V \subseteq Syntax$
  $\Rightarrow Increasing$
  $(V \lhd_o ExsSRO\ (W,\ \$\in_v))$
  $(EssSRO\ (V,\ Sf\ V\ \$\in_v))$
  $CombI$

essence_exstentional_lemma2

$\vdash \forall\ V\ W\ \$\in_v$
- $V \subseteq W \wedge V \subseteq Syntax$
  $\Rightarrow Increasing$
  $(W \lhd_o ExsSRO\ (W,\ \$\in_v))$
  $(EssSRO\ (V,\ Sf\ V\ \$\in_v))$
  $CombI$

totover_lemma

$\vdash \forall\ V\ r$
- $TotalOver\ V\ r$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in V \wedge y \in V \Rightarrow r\ x\ y = fTrue \vee r\ x\ y = fFalse)$

totover_true_lemma

$\vdash \forall\ V\ r$
- $TotalOver\ V\ r$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in V \wedge y \in V$
  $\Rightarrow (fTrue \leqslant_{t4} r\ x\ y \Leftrightarrow r\ x\ y = fTrue))$

totover_false_lemma

$\vdash \forall\ V\ r$
- $TotalOver\ V\ r$
  $\Rightarrow (\forall\ x\ y$
  $\bullet\ x \in V \wedge y \in V$
  $\Rightarrow (fFalse \leqslant_{t4} r\ x\ y \Leftrightarrow r\ x\ y = fFalse))$

BEqqRel_exs_lemma

$\vdash \forall\ V\ r\ x\ y$
- $x \in V \wedge y \in V \wedge BEqqRel\ V\ r\ x\ y$
  $\Rightarrow (r\ x\ x \Leftrightarrow r\ y\ y) \wedge (r\ x\ y \Leftrightarrow r\ y\ x)$

beqrel_equiv_lemma

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

beqqrel_equiv_lemma

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

booleqrel_equiv_lemma

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

booleqqrel_equiv_lemma

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

pre_tot_ext_lemma

$\vdash \forall\ V\ pr$
$\bullet\ PreExtensional\ V\ pr \wedge TotalOver\ V\ pr$
$\Rightarrow TotExtensional\ V\ (BoolRel\ pr)$

pre2_tot_ext_lemma

$\vdash \forall\ V\ pr$
$\bullet\ PreExtensional2\ V\ pr \wedge TotalOver\ V\ pr$
$\Rightarrow TotExtensional2\ V\ (BoolRel\ pr)$

equivclass_mem_lemma

$\vdash \forall\ V\ r\ x$
$\bullet\ x \in EquivClass\ (V,\ BEqRel\ V\ (BoolRel\ r))\ z$
$\Leftrightarrow x \in V$
$\wedge\ (\forall\ y$
$\bullet\ y \in V \Rightarrow (BoolRel\ r\ y\ x \Leftrightarrow BoolRel\ r\ y\ z))$

equivclass_mem_lemma2

$\vdash \forall\ V\ r\ x$
$\bullet\ x \in EquivClass\ (V,\ BEqqRel\ V\ (BoolRel\ r))\ z$
$\Leftrightarrow x \in V$
$\wedge\ (\forall\ y$
$\bullet\ y \in V$
$\Rightarrow (BoolRel\ r\ y\ x \Leftrightarrow BoolRel\ r\ y\ z)$
$\wedge\ (BoolRel\ r\ x\ y \Leftrightarrow BoolRel\ r\ z\ y))$

rep_independence_lemma

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r$
$\Rightarrow (\forall\ s\ t$
$\bullet\ s \in V \wedge t \in V$
$\Rightarrow (let\ u = EquivClass\ (V,\ BoolEqqRel\ V\ r)\ s$
$and\ v = EquivClass\ (V,\ BoolEqqRel\ V\ r)\ t$
$in\ (\forall\ x\ y \bullet\ x \in u \wedge y \in v \Rightarrow BoolRel\ r\ x\ y)$
$\Leftrightarrow (\exists\ x\ y$
$\bullet\ x \in u \wedge y \in v \wedge BoolRel\ r\ x\ y)))$

totpre_rep_independence_lemma

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow (\forall\ s\ t$
$\bullet\ s \in V \wedge t \in V$
$\Rightarrow (let\ u = EquivClass\ (V,\ BoolEqRel\ V\ r)\ s$
$and\ v = EquivClass\ (V,\ BoolEqRel\ V\ r)\ t$
$in\ (\forall\ x\ y \bullet\ x \in u \wedge y \in v \Rightarrow BoolRel\ r\ x\ y)$
$\Leftrightarrow (\exists\ x\ y$
$\bullet\ x \in u \wedge y \in v \wedge BoolRel\ r\ x\ y)))$

totpre_rep_independence_lemma2

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow (\forall\ s\ t\ u\ v$

$\bullet\ s \in V$
$\wedge\ t \in V$
$\wedge\ u \in V$
$\wedge\ v \in V$
$\wedge\ BoolEqRel\ V\ r\ s\ u$
$\wedge\ BoolEqRel\ V\ r\ t\ v$
$\Rightarrow (BoolRel\ r\ s\ t \Leftrightarrow BoolRel\ r\ u\ v))$

rep_independence_lemma2

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r$
$\Rightarrow (\forall\ s\ t\ u\ v$
$\bullet\ s \in V$
$\wedge\ t \in V$
$\wedge\ u \in V$
$\wedge\ v \in V$
$\wedge\ BoolEqqRel\ V\ r\ s\ u$
$\wedge\ BoolEqqRel\ V\ r\ t\ v$
$\Rightarrow (BoolRel\ r\ s\ t \Leftrightarrow BoolRel\ r\ u\ v))$

preext_ext_lemma

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow extensional\ (MSfromSFF\ (V,\ r))$

eq_eq_eqq_lemma

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow (\forall\ x\ y$
$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (BEqRel\ V\ (BoolRel\ r)\ x\ y$
$\Leftrightarrow BEqqRel\ V\ (BoolRel\ r)\ x\ y))$

msfromsff_eq_lemma

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow MSfromSFF\ (V,\ r) = MSfromSFF2\ (V,\ r)$

preext_ext_lemma2

$\vdash \forall\ V\ r$
$\bullet\ TotalOver\ V\ r \wedge PreExtensional\ V\ r$
$\Rightarrow extensional\ (MSfromSFF2\ (V,\ r))$

extsem_sf_thm

$\vdash \forall\ V \bullet\ V \subseteq SetReps \Rightarrow ExtSem\ V\ Sf$

compext_lemma1

$\vdash \forall\ V$
$\bullet\ V \subseteq SetReps$
$\Rightarrow (\forall\ r\ x\ y$
$\bullet\ x \in V \wedge y \in V$
$\Rightarrow (CompExt\ V\ r\ x\ y$
$\Leftrightarrow (\exists\ xt$
$\bullet\ \neg\ OverDefinedL\ V\ xt$
$\wedge\ IsUb$
$(PwS\ V\ \$\leqslant_{t4})$
$\{Extension\ r\ x;\ Extension\ r\ y\}$
$xt)))$

compext_lemma1b

$$\vdash \forall\ V \bullet V \subseteq Syntax \Rightarrow (\forall\ r\ x\ y \bullet x \in V \wedge y \in V \Rightarrow (CompExt\ V\ r\ x\ y \Leftrightarrow (\exists\ xt \bullet \neg\ OverDefinedL\ V\ xt \wedge IsUb\ (PwS\ V\ \$\leqslant_{t4})\ \{Extension\ r\ x;\ Extension\ r\ y\}\ xt)))$$

compext_lemma2

$$\vdash \forall\ V \bullet V \subseteq SetReps \Rightarrow (\forall\ \$\in_v\ x\ y \bullet x \in V \wedge y \in V \Rightarrow (CompExt\ V\ \$\in_v\ x\ y \Leftrightarrow (\exists\ xt \bullet \neg\ OverDefinedL\ V\ xt \wedge IsLub\ (PwS\ V\ \$\leqslant_{t4})\ \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\}\ xt)))$$

compext_lemma2b

$$\vdash \forall\ V \bullet V \subseteq Syntax \Rightarrow (\forall\ \$\in_v\ x\ y \bullet x \in V \wedge y \in V \Rightarrow (CompExt\ V\ \$\in_v\ x\ y \Leftrightarrow (\exists\ xt \bullet \neg\ OverDefinedL\ V\ xt \wedge IsLub\ (PwS\ V\ \$\leqslant_{t4})\ \{Extension\ \$\in_v\ x;\ Extension\ \$\in_v\ y\}\ xt)))$$

compess_lemma1

$$\vdash \forall\ V \bullet V \subseteq SetReps \Rightarrow (\forall\ r\ x\ y \bullet x \in V \wedge y \in V \Rightarrow (CompEss\ V\ r\ x\ y \Leftrightarrow (\exists\ es \bullet \neg\ OverDefinedL\ V\ es \wedge IsUb\ (PwS\ V\ \$\leqslant_{t4})\ \{Essence\ r\ x;\ Essence\ r\ y\}\ es)))$$

compess_lemma1b

$$\vdash \forall\ V \bullet V \subseteq Syntax$$

$\Rightarrow$ ($\forall$ r x y
• $x \in V \wedge y \in V$
$\Rightarrow$ ($CompEss$ V r x y
$\Leftrightarrow$ ($\exists$ es
• $\neg$ $OverDefinedL$ V es
$\wedge$ $IsUb$
(PwS V $\$\leqslant_{t4}$)
{$Essence$ r x; $Essence$ r y}
es)))

compess_lemma2

$\vdash \forall$ V
• $V \subseteq SetReps$
$\Rightarrow$ ($\forall$ $\$\in_v$ x y
• $x \in V \wedge y \in V$
$\Rightarrow$ ($CompEss$ V $\$\in_v$ x y
$\Leftrightarrow$ ($\exists$ es
• $\neg$ $OverDefinedL$ V es
$\wedge$ $IsLub$
(PwS V $\$\leqslant_{t4}$)
{$Essence$ $\$\in_v$ x; $Essence$ $\$\in_v$ y}
es)))

compess_lemma2b

$\vdash \forall$ V
• $V \subseteq Syntax$
$\Rightarrow$ ($\forall$ $\$\in_v$ x y
• $x \in V \wedge y \in V$
$\Rightarrow$ ($CompEss$ V $\$\in_v$ x y
$\Leftrightarrow$ ($\exists$ es
• $\neg$ $OverDefinedL$ V es
$\wedge$ $IsLub$
(PwS V $\$\leqslant_{t4}$)
{$Essence$ $\$\in_v$ x; $Essence$ $\$\in_v$ y}
es)))

ExtUb_lemma1

$\vdash \forall$ (V, $\$\in_v$) x y
• $V \subseteq SetReps$
$\wedge \neg$ $OverDefined$ V $\$\in_v$
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $CompExt$ V $\$\in_v$ x y
$\Rightarrow \neg$ $OverDefinedL$ V ($ExtUb$ (V, $\$\in_v$) x y)
$\wedge$ ($\forall$ z
• $z \in V$
$\Rightarrow$ (z $\in_v$ x) $\leqslant_{t4}$ $ExtUb$ (V, $\$\in_v$) x y z
$\wedge$ (z $\in_v$ y) $\leqslant_{t4}$ $ExtUb$ (V, $\$\in_v$) x y z)

ExtLub_lemma1

$\vdash \forall$ (V, $\$\in_v$) x y
• $V \subseteq SetReps$
$\wedge \neg$ $OverDefined$ V $\$\in_v$
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $CompExt$ V $\$\in_v$ x y

$$\Rightarrow \neg\ OverDefinedL\ V\ (ExtLub\ (V,\ \$\in_v)\ x\ y)$$
$$\wedge\ (\forall\ z$$
$$\bullet\ z \in V$$
$$\Rightarrow (z \in_v x) \leqslant_{t4} ExtLub\ (V,\ \$\in_v)\ x\ y\ z$$
$$\wedge\ (z \in_v y) \leqslant_{t4} ExtLub\ (V,\ \$\in_v)\ x\ y\ z)$$

ExtLub2_lemma1

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq Syntax$$
$$\wedge\ \neg\ OverDefined\ V\ \$\in_v$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ CompExt\ V\ \$\in_v\ x\ y$$
$$\Rightarrow \neg\ OverDefinedL\ V\ (ExtLub2\ (V,\ \$\in_v)\ x\ y)$$
$$\wedge\ (\forall\ z$$
$$\bullet\ z \in V$$
$$\Rightarrow (z \in_v x) \leqslant_{t4} ExtLub2\ (V,\ \$\in_v)\ x\ y\ z$$
$$\wedge\ (z \in_v y)$$
$$\leqslant_{t4} ExtLub2\ (V,\ \$\in_v)\ x\ y\ z)$$

EssUb_lemma1

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq SetReps$$
$$\wedge\ \neg\ OverDefined\ V\ \$\in_v$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ CompEss\ V\ \$\in_v\ x\ y$$
$$\Rightarrow \neg\ OverDefinedL\ V\ (EssUb\ (V,\ \$\in_v)\ x\ y)$$
$$\wedge\ (\forall\ z$$
$$\bullet\ z \in V$$
$$\Rightarrow (x \in_v z) \leqslant_{t4} EssUb\ (V,\ \$\in_v)\ x\ y\ z$$
$$\wedge\ (y \in_v z) \leqslant_{t4} EssUb\ (V,\ \$\in_v)\ x\ y\ z)$$

EssLub_lemma1

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq SetReps$$
$$\wedge\ \neg\ OverDefined\ V\ \$\in_v$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ CompEss\ V\ \$\in_v\ x\ y$$
$$\Rightarrow \neg\ OverDefinedL\ V\ (EssLub\ (V,\ \$\in_v)\ x\ y)$$
$$\wedge\ (\forall\ z$$
$$\bullet\ z \in V$$
$$\Rightarrow (x \in_v z) \leqslant_{t4} EssLub\ (V,\ \$\in_v)\ x\ y\ z$$
$$\wedge\ (y \in_v z) \leqslant_{t4} EssLub\ (V,\ \$\in_v)\ x\ y\ z)$$

EssLub_lemma1b

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y$$
$$\bullet\ V \subseteq Syntax$$
$$\wedge\ \neg\ OverDefined\ V\ \$\in_v$$
$$\wedge\ x \in V$$
$$\wedge\ y \in V$$
$$\wedge\ CompEss\ V\ \$\in_v\ x\ y$$
$$\Rightarrow \neg\ OverDefinedL\ V\ (EssLub2\ (V,\ \$\in_v)\ x\ y)$$
$$\wedge\ (\forall\ z$$
$$\bullet\ z \in V$$

$$\Rightarrow (x \in_v z) \leqslant_{t4} \mathit{EssLub2}\ (V, \$\in_v)\ x\ y\ z$$
$$\wedge\ (y \in_v z)$$
$$\leqslant_{t4} \mathit{EssLub2}\ (V, \$\in_v)\ x\ y\ z)$$

$PmrEq\_MsExtend\_lemma1$

$$\vdash \forall\ (V,\ r)\ x\ y$$
$$\bullet\ V \subseteq \mathit{Syntax} \Rightarrow \mathit{PmrEq}\ V\ r\ (\mathit{MsExtend}\ (V,\ r)\ x\ y)$$

$PmrEq\_MsExtend\_lemma2$

$$\vdash \forall\ (V,\ r)\ x\ y$$
$$\bullet\ V \subseteq \mathit{Syntax} \Rightarrow \mathit{PmrEq}\ V\ (\mathit{MsExtend}\ (V,\ r)\ x\ y)\ r$$

$PmrEq\_MsExtend2\_lemma2$

$$\vdash \forall\ (V,\ r)\ x\ y$$
$$\bullet\ V \subseteq \mathit{Syntax} \Rightarrow \mathit{PmrEq}\ V\ (\mathit{MsExtend2}\ (V,\ r)\ x\ y)\ r$$

$EvalForm\_MsExtend\_lemma1$

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\ z$$
$$\bullet\ V \subseteq \mathit{Syntax} \wedge y \in V \wedge z \in V$$
$$\Rightarrow \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend}\ (V,\ r)\ x\ y)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ y)$$
$$= \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ y)$$

$EvalForm\_MsExtend2\_lemma1$

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\ z$$
$$\bullet\ V \subseteq \mathit{Syntax} \wedge y \in V \wedge z \in V$$
$$\Rightarrow \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ y)$$
$$= \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ y)$$

$EvalForm\_MsExtend\_lemma2$

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\ z$$
$$\bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge z \in V$$
$$\Rightarrow \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend}\ (V,\ r)\ x\ y)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ x)$$
$$= \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ r)$$
$$z$$
$$(\mathit{Param\_}\varnothing\ x)$$

$EvalForm\_MsExtend2\_lemma2$

$$\vdash \forall\ (V,\ \$\in_v)\ x\ y\ z$$
$$\bullet\ V \subseteq \mathit{Syntax} \wedge x \in V \wedge z \in V$$
$$\Rightarrow \mathit{EvalForm}$$
$$(\mathit{EvalCf\_ftv},\ \$\leqslant_{t4},\ V,\ \mathit{MsExtend2}\ (V,\ r)\ x\ y)$$
$$z$$

$(Param\_\varnothing\ x)$
$= EvalForm$
$(EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ r)$
z
$(Param\_\varnothing\ x)$

$EvalForm\_MsExtend2\_lemma3$

$\vdash\ \forall\ (V,\ \$\in_v)\ x\ y\ v\ w$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge v \in V \wedge w \in V$
  $\Rightarrow EvalForm$
  $(EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ MsExtend2\ (V,\ r)\ x\ y)$
  v
  $(Param\_\varnothing\ w)$
  $= EvalForm$
  $(EvalCf\_ftv,\ \$\leqslant_{t4},\ V,\ r)$
  v
  $(Param\_\varnothing\ w)$

$Sf\_MsExtend2\_lemma1$

$\vdash\ \forall\ (V,\ \$\in_v)\ x\ y\ v\ w$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge v \in V \wedge w \in V$
  $\Rightarrow Sf\ V\ (MsExtend2\ (V,\ \$\in_v)\ x\ y)\ v\ w$
  $= Sf\ V\ \$\in_v\ v\ w$

$MsExtend2\_\varnothing_g\_Lub\_lemma1$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$
- $V \subseteq Syntax$
  $\wedge\ CompExt\ V\ r\ x\ y$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ z \in V$
  $\Rightarrow MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g$
  $= Lub$
  $\$\leqslant_{t4}$
  $\{MsExtend2\ (V,\ r)\ x\ y\ z\ x;$
  $MsExtend2\ (V,\ r)\ x\ y\ z\ y\}$

$MsExtend2\_\varnothing_g\_Lub\_lemma2$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$
- $V \subseteq Syntax$
  $\wedge\ CompEss\ V\ r\ x\ y$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ z \in V$
  $\Rightarrow MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z$
  $= Lub$
  $\$\leqslant_{t4}$
  $\{MsExtend2\ (V,\ r)\ x\ y\ x\ z;$
  $MsExtend2\ (V,\ r)\ x\ y\ y\ z\}$

$MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma1$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$
- $V \subseteq Syntax$
  $\wedge\ CompEss\ V\ r\ x\ y$
  $\wedge\ x \in V$
  $\wedge\ y \in V$

$\wedge\ z \in V$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ x\ z$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z$

$MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma1b$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge\ Compatible\ V\ r\ x\ y$

$\wedge\ x \in V$

$\wedge\ y \in V$

$\wedge\ z \in V \cup \{\varnothing_g\}$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ x\ z$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z$

$MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma1$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge\ CompEss\ V\ r\ x\ y$

$\wedge\ x \in V$

$\wedge\ y \in V$

$\wedge\ z \in V$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ y\ z$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z$

$MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma1b$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge\ Compatible\ V\ r\ x\ y$

$\wedge\ x \in V$

$\wedge\ y \in V$

$\wedge\ z \in V \cup \{\varnothing_g\}$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ y\ z$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ \varnothing_g\ z$

$MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma2$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge\ CompExt\ V\ r\ x\ y$

$\wedge\ x \in V$

$\wedge\ y \in V$

$\wedge\ z \in V$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ z\ x$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g$

$MsExtend2\_y\_\leqslant_{t4}\_\varnothing_g\_lemma2$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge\ CompExt\ V\ r\ x\ y$

$\wedge\ x \in V$

$\wedge\ y \in V$

$\wedge\ z \in V$

$\Rightarrow\ MsExtend2\ (V,\ r)\ x\ y\ z\ y$

$\leqslant_{t4}\ MsExtend2\ (V,\ r)\ x\ y\ z\ \varnothing_g$

$MsExtend2\_x\_\leqslant_{t4}\_\varnothing_g\_lemma2b$

$\vdash\ \forall\ (V,\ r)\ x\ y\ z$

- $V \subseteq Syntax$

$\wedge$ *CompEss* V r x y
$\wedge$ *CompExt* V r x y
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $z \in V \cup \{\varnothing_g\}$
$\Rightarrow$ *MsExtend2* (V, r) x y z x
$\leqslant_{t4}$ *MsExtend2* (V, r) x y z $\varnothing_g$

MsExtend2_y_$\leqslant_{t4}$_$\varnothing_g$_lemma2b

$\vdash \forall (V, r)$ x y z
• $V \subseteq$ *Syntax*
$\wedge$ *CompEss* V r x y
$\wedge$ *CompExt* V r x y
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $z \in V \cup \{\varnothing_g\}$
$\Rightarrow$ *MsExtend2* (V, r) x y z y
$\leqslant_{t4}$ *MsExtend2* (V, r) x y z $\varnothing_g$

MsExtend_¬OverDefined_lemma

$\vdash \forall (V, \$\in_v)$ x y
• $V \subseteq$ *Syntax*
$\wedge$ $\neg$ *OverDefined* V $\$\in_v$
$\wedge$ *CompEss* V $\$\in_v$ x y
$\wedge$ *CompExt* V $\$\in_v$ x y
$\Rightarrow \neg$ *OverDefined* V (*MsExtend* $(V, \$\in_v)$ x y)

MsExtend2_¬OverDefined_lemma

$\vdash \forall (V, \$\in_v)$ x y
• $V \subseteq$ *Syntax*
$\wedge$ $\neg$ *OverDefined* V $\$\in_v$
$\wedge$ *CompEss* V $\$\in_v$ x y
$\wedge$ *CompExt* V $\$\in_v$ x y
$\Rightarrow \neg$ *OverDefined* V (*MsExtend2* $(V, \$\in_v)$ x y)

MsExtend_¬OverDefined_lemma2

$\vdash \forall (V, \$\in_v)$ x y
• $V \subseteq$ *SetReps*
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $\neg$ *OverDefined* V $\$\in_v$
$\wedge$ *CompEss* V $\$\in_v$ x y
$\wedge$ *CompExt* V $\$\in_v$ x y
$\Rightarrow (\forall$ v
• $v \in V \Rightarrow \neg$ *MsExtend* $(V, \$\in_v)$ x y v $\varnothing_g = fT$)

MsExtend2_¬OverDefined_lemma2

$\vdash \forall (V, \$\in_v)$ x y
• $V \subseteq$ *Syntax*
$\wedge$ $x \in V$
$\wedge$ $y \in V$
$\wedge$ $\neg$ *OverDefined* V $\$\in_v$
$\wedge$ *CompEss* V $\$\in_v$ x y
$\wedge$ *CompExt* V $\$\in_v$ x y
$\Rightarrow (\forall$ v
• $v \in V \Rightarrow \neg$ *MsExtend2* $(V, \$\in_v)$ x y v $\varnothing_g = fT$)

MsExtend_¬OverDefined_lemma3

$$
\begin{array}{l}
\vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \wedge\ x \in V \\
\qquad \wedge\ y \in V \\
\qquad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\qquad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\qquad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow (\forall\ v \\
\qquad \bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y\ \varnothing_g\ v = fT)
\end{array}
$$

MsExtend2_¬OverDefined_lemma3

$$
\begin{array}{l}
\vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad \bullet\ V \subseteq \mathit{Syntax} \\
\qquad \wedge\ x \in V \\
\qquad \wedge\ y \in V \\
\qquad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\qquad \wedge\ \mathit{CompEss}\ V\ \$\in_v\ x\ y \\
\qquad \wedge\ \mathit{CompExt}\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow (\forall\ v \\
\qquad \bullet\ v \in V \Rightarrow \neg\ \mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y\ \varnothing_g\ v = fT)
\end{array}
$$

MsExtend_¬OverDefined_lemma4

$$
\begin{array}{l}
\vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \wedge\ x \in V \\
\qquad \wedge\ y \in V \\
\qquad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\qquad \wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow \neg\ \mathit{OverDefined} \\
\qquad\quad (V\ \cup\ \{\varnothing_g\}) \\
\qquad\quad (\mathit{MsExtend}\ (V,\ \$\in_v)\ x\ y)
\end{array}
$$

MsExtend2_¬OverDefined_lemma4

$$
\begin{array}{l}
\vdash \forall\ (V,\ \$\in_v)\ x\ y \\
\quad \bullet\ V \subseteq \mathit{Syntax} \\
\qquad \wedge\ x \in V \\
\qquad \wedge\ y \in V \\
\qquad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\qquad \wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\qquad \Rightarrow \neg\ \mathit{OverDefined} \\
\qquad\quad (V\ \cup\ \{\varnothing_g\}) \\
\qquad\quad (\mathit{MsExtend2}\ (V,\ \$\in_v)\ x\ y)
\end{array}
$$

ConWkStrictSf_lemma

$$\vdash \forall\ V \bullet\ V \subseteq \mathit{SetReps} \Rightarrow \mathit{ConWkStrict}\ V\ \mathit{Sf}$$

ConCoWkStrictSf_lemma

$$\vdash \forall\ V \bullet\ V \subseteq \mathit{Syntax} \Rightarrow \mathit{ConCoWkStrict}\ V\ \mathit{Sf}$$

ODE_SF_lemma

$$
\begin{array}{l}
\vdash \forall\ V\ \$\in_v \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \wedge\ \mathit{OverDefinedEss} \\
\qquad\quad (V\ \cup\ \{\varnothing_g\}) \\
\qquad\quad V \\
\qquad\quad (\mathit{Sf}\ (V\ \cup\ \{\varnothing_g\})\ \$\in_v) \\
\qquad \Rightarrow \mathit{OverDefined}\ (V\ \cup\ \{\varnothing_g\})\ \$\in_v
\end{array}
$$

ODE_SF_lemma2

$$\vdash \forall\ V\ \$\in_v$$
- $V \subseteq SetReps$
  $\wedge\ OverDefinedEss\ (V \cup \{\varnothing_g\})\ V\ (Sf\ V\ \$\in_v)$
  $\Rightarrow OverDefined\ (V \cup \{\varnothing_g\})\ \$\in_v$

ODE_SF_MsExtend2_lemma

$$\vdash \forall\ V\ r\ x\ y$$
- $V \subseteq SetReps$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ Compatible\ V\ r\ x\ y$
  $\wedge\ OverDefinedEss$
  $(V \cup \{\varnothing_g\})$
  V
  $(Sf\ (V \cup \{\varnothing_g\})\ (MsExtend2\ (V,\ r)\ x\ y))$
  $\Rightarrow OverDefined\ V\ r$

ODE_SF_MsExtend2_lemma2

$$\vdash \forall\ V\ r\ x\ y$$
- $V \subseteq SetReps$
  $\wedge\ x \in V$
  $\wedge\ y \in V$
  $\wedge\ Compatible\ V\ r\ x\ y$
  $\wedge\ OverDefinedEss$
  $(V \cup \{\varnothing_g\})$
  V
  $(Sf\ V\ (MsExtend2\ (V,\ r)\ x\ y))$
  $\Rightarrow OverDefined\ V\ r$

ExsSRO_MsExtend2_lemma1

$$\vdash \forall\ V\ r\ x\ y$$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge Compatible\ V\ r\ x\ y$
  $\Rightarrow ExsSRO\ (V,\ MsExtend2\ (V,\ r)\ x\ y)\ x\ \varnothing_g$
  $\wedge\ ExsSRO\ (V,\ MsExtend2\ (V,\ r)\ x\ y)\ y\ \varnothing_g$

ExsSRO_MsExtend2_lemma2

$$\vdash \forall\ V\ r\ x\ y$$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge Compatible\ V\ r\ x\ y$
  $\Rightarrow ExsSRO\ (V \cup \{\varnothing_g\},\ MsExtend2\ (V,\ r)\ x\ y)\ x\ \varnothing_g$
  $\wedge\ ExsSRO$
  $(V \cup \{\varnothing_g\},\ MsExtend2\ (V,\ r)\ x\ y)$
  y
  $\varnothing_g$

Compatibility_lemma1

$$\vdash \forall\ V\ \$\in_v\ x\ y$$
- $V \subseteq Syntax \wedge x \in V \wedge y \in V \wedge Compatible\ V\ \$\in_v\ x\ y$
  $\Rightarrow EssSRO$
  $(V,\ Sf\ V\ (MsExtend2\ (V,\ \$\in_v)\ x\ y))$
  x
  $\varnothing_g$
  $\wedge\ EssSRO$
  $(V,\ Sf\ V\ (MsExtend2\ (V,\ \$\in_v)\ x\ y))$
  y
  $\varnothing_g$

Compatibility_lemma2

$$
\begin{array}{l}
\vdash \forall\ V\ \$\in_v\ x\ y \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \wedge\ x \in V \\
\qquad \wedge\ y \in V \\
\qquad \wedge\ \mathit{Compatible}\ V\ \$\in_v\ x\ y \\
\qquad \wedge\ \neg\ \mathit{OverDefined}\ V\ \$\in_v \\
\qquad \Rightarrow \mathit{CompEss}\ V\ (\mathit{Sf}\ V\ \$\in_v)\ x\ y
\end{array}
$$

CompExtSem_Sf_thm

$$
\vdash \forall\ V \bullet\ V \subseteq \mathit{SetReps} \Rightarrow \mathit{CompExtSem}\ V\ \mathit{Sf}
$$

sfchain_induction_thm

$$
\begin{array}{l}
\vdash \forall\ V\ p \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \wedge\ (\forall\ x \\
\qquad \bullet\ x \in \mathit{SfChain}\ V \\
\qquad\quad \wedge\ (\forall\ y \bullet\ y \in \mathit{SfChain}\ V\ \wedge\ y \leqslant_\in x \Rightarrow p\ y) \\
\qquad\quad \Rightarrow p\ (\mathit{Sf}\ V\ x)) \\
\qquad \wedge\ (\forall\ x \\
\qquad \bullet\ x \in \mathit{SfChain}\ V \\
\qquad\quad \wedge\ x \\
\qquad\quad\ = \mathit{Lub} \\
\qquad\quad\ \$\leqslant_\in \\
\qquad\quad\ \{y \\
\qquad\quad\ \ | y \in \mathit{SfChain}\ V \\
\qquad\quad\ \ \wedge\ y \leqslant_\in x \\
\qquad\quad\ \ \wedge\ \neg\ x \leqslant_\in y\} \\
\qquad\quad \wedge\ (\forall\ y \\
\qquad\quad \bullet\ y \in \mathit{SfChain}\ V\ \wedge\ y \leqslant_\in x\ \wedge\ \neg\ x \leqslant_\in y \\
\qquad\quad\ \ \Rightarrow p\ y) \\
\qquad\quad \Rightarrow p\ x) \\
\qquad \Rightarrow (\forall\ x \bullet\ x \in \mathit{SfChain}\ V \Rightarrow p\ x)
\end{array}
$$

sfchain_mono_thm

$$
\vdash \forall\ V\ x \bullet\ x \in \mathit{SfChain}\ V \Rightarrow x \leqslant_\in \mathit{Sf}\ V\ x
$$

sfchain_linear_lemma

$$
\vdash \forall\ V \bullet\ \mathit{LinearOrder}\ (\mathit{SfChain}\ V,\ \$\leqslant_\in)
$$

sfsubchain_linear_lemma

$$
\vdash \forall\ V\ X \bullet\ X \subseteq \mathit{SfChain}\ V \Rightarrow \mathit{LinearOrder}\ (X,\ \$\leqslant_\in)
$$

sfchain_¬overdefined_lemma

$$
\begin{array}{l}
\vdash \forall\ V \\
\quad \bullet\ V \subseteq \mathit{SetReps} \\
\qquad \Rightarrow (\forall\ x \bullet\ x \in \mathit{SfChain}\ V \Rightarrow \neg\ \mathit{OverDefined}\ V\ x)
\end{array}
$$

6 INDEX