Set Theory as Consistent Infinitary Comprehension

Roger Bishop Jones

Abstract

This paper is concerned with set theory conceived as a maximal consistent theory of set comprehension. This is interpreted by looking for large subdomains of a notation for infinitary comprehension, and the theory is developed from such interpretations.

Created: 2006/11/29

Last Change Date: 2012/08/11 21:01:53

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

Id: t021.doc,v 1.16 2012/08/11 21:01:53 rbj Exp

Contents

To Do

-
-

References

[1] Thomas Jech. *Set Theory*. Springer Verlag, 2002.

1 INTRODUCTION

The idea is to come up with a set theory whose subject matter combines the well-founded sets of the cumulative heirarchy with a similarly rich non-well-founded ontology.

The well-founded ontology is based on a simple ontological principle embodied in the following informal definition (transfinite, inductive) of "well-founded set":

a well-founded set is any definite collection of well-founded sets

Analysis and futher explication of this definition is worthwhile, but is not our present purpose, for which we will assume that the above description suffices and proceed to the non-well-founded sets.

There are different conceptions of well-founded-set from which we may chose, and I do not know of any single all encompassing conception. Probably the single most prominent conception comes from the idea of set comprehension, viz. that to some useful subset of the properties of sets there correspond sets which are the extensions of the properties. In the beginnings, Frege allowed unrestrained comprehension, which resulted in inconsistency. A variety of set theories may then be regarded as limitations on the ideal of unrestrained comprehension sufficient to avoid incoherence. One of these yields the well-founded sets (or some of them).

If the incoherence of unrestricted comprehension could be located in specific properties then sets correponding to their extension could be omitted from the the domain of discourse and a maximal consistent theory of set comprehension might result. There is however no obvious way to do this. The properties are themselves all equally coherent, it is only when we attempt to realise an ontology of sets whose extensions correpond to the properties that we run into trouble, and when this happens it is not so easy to say which properties are at fault. Nevertheless, this is what we are going to attempt here.

There is another problem which must be mentioned here. When we seek a full theory of comprehension, we are looking for a set theory in which the set properties of sets is the same as the set of sets itself. There is a problem of cardinality here, which forces us to limit our selection of properties. This problem we address by taking, instead of the notion of property in general, the notion of definable property or rule will be used. The number of such definable properties depends upon the language in which the definition takes place, and we arrange for the cardinality of the syntax of this language to be the same as the cardinality of the domain of our set theory (which will also be the cardinality of the well-founded part of the domain). This effect can be achieved by coding up the properties in an infinitary language whose syntax is made from well-founded sets. The language will have sufficent syntax for every well-founded set to correspond to a definable property as well as the non-well-founded sets, and we therefore by means of this infinitary syntax assimilate both non well-founded and well-founded sets, but also the notion of set as graph an set as rule.

The plan is:

- Define using the ontology of a well-founded set theory (GS) a notation for infinitary comprehension (this will be a transfinite inductive definition of a way of coding up set comprehension in an infinitary language for set theory).
- Give a semantics to this notation as a functor which transforms pairs of partial equality and membership relations over closed comprehensions.
- Find maximal subsets of the class of closed comprehensions over which the functor has a fixed point, and consider the theory which arises when these subsets with the fixedpoint semantics are taken as interpretations of set theory.

SML

```
open_theory "GS";
force_new_theory "ICsyn";
new_parent "U_orders";
force_new_pc "'ICsyn";
merge_pcs ["'savedthm_cs_∃_proof"] "'ICsyn";
set_merge_pcs ["hol1", "'GS1", "'ICsyn"];
```

2 MISCELLANEA

I am at present using a set theory based on an old version of the theory of well-foundedness and well-founded recursion.

Some definitions are supplied here which patch over the differences between the new and old treatments of well-foundedness.

2.1 Some Synonyms

$$\begin{array}{l} \boldsymbol{tc\_TranClsr\_thm} = \\ \quad \vdash \forall\ r\bullet\ (Universe,\ tc\ r) = TranClsr\ (Universe,\ r) \end{array}$$

3 SYNTAX

3.1 Informal Abstract Syntax

Here is an abstract syntax:

$$\begin{array}{lll} term & ::= & ordinal \\ & | & SetComp\ formula \\ & & \\ formula & ::= & term = term \\ & | & term \in term \\ & | & tf\ \ ordinal\ (\mathbb{P}\ formula) \end{array}$$

Note that the use of the powerset constructor on the right disqualifies this from being a normal recursive datatype. The apparent incoherence of having an equation with the powerset of the lhs occuring in the rhs is resolved by the powerset in question not being a HOL powerset, but a set of sets in a full blooded set theory. The definition defines two classes which are not sets, and because they are classes they have the same size as the class of their subsets and the cardinality problem goes away. A definition of this kind can be formalised as what Isabelle calls an inductive set definition bearing in mind here that the sets in question are HOL subsets of a type which is a class of sets. i.e. it is really an inductive class definition.

The ordinals here are used as variables, and Von Neumann ordinals are required. They are in fact used like De Bruijn indices, so comprehension always binds the variable zero at the outermost level, and binds a higher numbered variable in inner scopes. A single constructor is provided for truth functional expressions and is to be understood as the negation of the universal quantification of the conjunction of a set (of any cardinality) of formulae. This quantified simultaneously quantifies over any number of variables.

3.2 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.

HOL Constant

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

$$\forall v \bullet MkVar\ v = (Nat_g\ 0) \mapsto_g v$$

HOL Constant

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

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

HOL Constant

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

$$VarNum = snd$$

HOL Constant

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

$$\forall f \bullet MkComp\ f = (Nat_g\ 1) \mapsto_g f$$

HOL Constant

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

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

HOL Constant

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

$$CompBody = snd$$

HOL Constant

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

$$\forall lr \bullet MkEq\ lr = (Nat_g\ 2) \mapsto_g ((Fst\ lr) \mapsto_g (Snd\ lr))$$

HOL Constant

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

$$\forall t \bullet IsEq\ t = fst\ t = (Nat_g\ 2)$$

HOL Constant

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

$$\forall\ lr\bullet\ \ MkMem\ lr = (Nat_g\ 3) \mapsto_g ((Fst\ lr) \mapsto_g (Snd\ lr))$$

HOL Constant

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

$$\forall t\bullet\ IsMem\ t = fst\ t = (Nat_g\ 3)$$

HOL Constant

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

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

HOL Constant

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

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

HOL Constant

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

$$\forall vc\bullet\ MkTf\ vc = (Nat_g\ 4) \mapsto_g ((Fst\ vc) \mapsto_g (Snd\ vc))$$

HOL Constant

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

$$\forall t\bullet\ IsTf\ t = fst\ t = (Nat_g\ 4)$$

HOL Constant

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

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

HOL Constant

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

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

$$
\begin{array}{l}
\textbf{\textit{Is\_clauses}} = \\
\quad \vdash ((\forall\ v\bullet\ IsVar\ (MkVar\ v)) \\
\qquad \wedge\ (\forall\ f\bullet \neg\ IsVar\ (MkComp\ f)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsVar\ (MkEq\ lr)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsVar\ (MkMem\ lr)) \\
\qquad \wedge\ (\forall\ vc\bullet \neg\ IsVar\ (MkTf\ vc))) \\
\quad \wedge\ ((\forall\ v\bullet \neg\ IsComp\ (MkVar\ v)) \\
\qquad \wedge\ (\forall\ f\bullet\ IsComp\ (MkComp\ f)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsComp\ (MkEq\ lr)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsComp\ (MkMem\ lr)) \\
\qquad \wedge\ (\forall\ vc\bullet \neg\ IsComp\ (MkTf\ vc))) \\
\quad \wedge\ ((\forall\ v\bullet \neg\ IsEq\ (MkVar\ v)) \\
\qquad \wedge\ (\forall\ f\bullet \neg\ IsEq\ (MkComp\ f)) \\
\qquad \wedge\ (\forall\ lr\bullet\ IsEq\ (MkEq\ lr)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsEq\ (MkMem\ lr)) \\
\qquad \wedge\ (\forall\ vc\bullet \neg\ IsEq\ (MkTf\ vc))) \\
\quad \wedge\ ((\forall\ v\bullet \neg\ IsMem\ (MkVar\ v)) \\
\qquad \wedge\ (\forall\ f\bullet \neg\ IsMem\ (MkComp\ f)) \\
\qquad \wedge\ (\forall\ lr\bullet \neg\ IsMem\ (MkEq\ lr)) \\
\qquad \wedge\ (\forall\ lr\bullet\ IsMem\ (MkMem\ lr)) \\
\qquad \wedge\ (\forall\ vc\bullet \neg\ IsMem\ (MkTf\ vc))) \\
\quad \wedge\ (\forall\ v\bullet \neg\ IsTf\ (MkVar\ v)) \\
\quad \wedge\ (\forall\ f\bullet \neg\ IsTf\ (MkComp\ f)) \\
\quad \wedge\ (\forall\ lr\bullet \neg\ IsTf\ (MkEq\ lr)) \\
\quad \wedge\ (\forall\ lr\bullet \neg\ IsTf\ (MkMem\ lr)) \\
\quad \wedge\ (\forall\ vc\bullet\ IsTf\ (MkTf\ vc)) \\
\\
\textbf{\textit{Is\_not\_fc\_clauses}} = \\
\quad \vdash (\forall\ x\bullet\ IsVar\ x \Rightarrow \neg\ IsComp\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsMem\ x \wedge \neg\ IsTf\ x) \\
\qquad \wedge\ (\forall\ x\bullet\ IsComp\ x \Rightarrow \neg\ IsVar\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsMem\ x \wedge \neg\ IsTf\ x) \\
\qquad \wedge\ (\forall\ x\bullet\ IsEq\ x \Rightarrow \neg\ IsComp\ x \wedge \neg\ IsVar\ x \wedge \neg\ IsMem\ x \wedge \neg\ IsTf\ x) \\
\qquad \wedge\ (\forall\ x\bullet\ IsMem\ x \Rightarrow \neg\ IsComp\ x \wedge \neg\ IsVar\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsTf\ x) \\
\qquad \wedge\ (\forall\ x\bullet\ IsTf\ x \Rightarrow \neg\ IsComp\ x \wedge \neg\ IsVar\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsMem\ x)
\end{array}
$$

Some derived syntax:

HOL Constant

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

HOL Constant

$$
\begin{array}{l}
\textbf{\textit{IsTerm}} : GS \to BOOL \\
\hline
\quad \forall t\bullet\ IsTerm\ t \Leftrightarrow IsVar\ t \vee IsComp\ t
\end{array}
$$

HOL Constant

$$\boldsymbol{Terms} : GS\ SET \to GS\ SET$$

$$\forall s \bullet\ Terms\ s = \{x \mid x \in s \wedge IsTerm\ x\}$$

$$\boldsymbol{terms\_mono\_thm} =$$
$$\vdash \forall\ s\ t \bullet\ s \subseteq t \Rightarrow Terms\ s \subseteq Terms\ t$$

HOL Constant

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

$$\forall x \bullet\ IsForm\ x \Leftrightarrow IsMem\ x \vee IsEq\ x \vee IsTf\ x$$

HOL Constant

$$\boldsymbol{Formulas} : GS\ SET \to GS\ SET$$

$$\forall s \bullet\ Formulas\ s = \{x \mid x \in s \wedge IsForm\ x\}$$

$$\boldsymbol{formulas\_mono\_thm} =$$
$$\vdash \forall\ s\ t \bullet\ s \subseteq t \Rightarrow Formulas\ s \subseteq Formulas\ t$$

3.3 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

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

$$\begin{aligned}
&\forall\ s \bullet\ RepClosed\ s \Leftrightarrow \\
&\quad (\forall\ ord \bullet\ \ ordinal\ ord \Rightarrow MkVar\ ord \in s) \\
\wedge\ &\quad (\forall\ f \bullet\ f \in Formulas\ s \Rightarrow MkComp\ f \in s) \\
\wedge\ &\quad (\forall\ t1\ t2 \bullet\ t1 \in Terms\ s \wedge t2 \in Terms\ s \\
&\qquad \Rightarrow MkEq\ (t1, t2) \in s \\
&\qquad \wedge MkMem\ (t1, t2) \in s) \\
\wedge\ &\quad (\forall\ ord\ fs \bullet\ ordinal\ ord \wedge X_g\ fs \subseteq Formulas\ s \\
&\qquad \Rightarrow MkTf\ (ord, fs) \in s)
\end{aligned}$$

The "Rep" in "RepClosed" stands for representatives, the idea being that we are defining a syntax for objects which represent sets (and will later assign a semantics to these representatives as sets of representatives). As it happens it is only the comprehensions which represent sets.

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

HOL Constant

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

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

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

$$\boldsymbol{repclosed\_term\_thm} =$$
$$\vdash \forall\ t\bullet\ RepClosed\ t \Rightarrow Terms\ Syntax \subseteq Terms\ t$$

$$\boldsymbol{repclosed\_forn\_thm} =$$
$$\vdash \forall\ t\bullet\ RepClosed\ t \Rightarrow Formulas\ Syntax \subseteq Formulas\ t$$

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

Informally these should say:

- Everything in Syntax is either a Term or a Formula.
- Syntax is closed under the five syntactic constructors.
- The syntax constructors are all 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).

$$\boldsymbol{repclosed\_syntax\_lemma} =$$
$$\vdash RepClosed\ Syntax$$

$$\boldsymbol{repclosed\_syntax\_thm} =$$
$$\begin{aligned}
&\vdash (\forall\ ord\bullet\ ordinal\ ord \Rightarrow MkVar\ ord \in Syntax)\\
&\quad \wedge (\forall\ f\bullet\ f \in Formulas\ Syntax \Rightarrow MkComp\ f \in Syntax)\\
&\quad \wedge (\forall\ t1\ t2\\
&\quad \bullet\ t1 \in Terms\ Syntax \wedge t2 \in Terms\ Syntax\\
&\qquad \Rightarrow MkEq\ (t1, t2) \in Syntax \wedge MkMem\ (t1, t2) \in Syntax)\\
&\quad \wedge (\forall\ ord\ fs\\
&\quad \bullet\ ordinal\ ord \wedge (\forall\ x\bullet\ x \in X_g\ fs \Rightarrow x \in Formulas\ Syntax)\\
&\qquad \Rightarrow MkTf\ (ord, fs) \in Syntax)
\end{aligned}$$

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

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

We need to be able to define functions by recursion over the syntax of comprehensions. For this we need a recursion theorem. We have a recursion theorem for well founded recursion already, so we can

build on that. To use that recursion theorem 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 for the syntax of comprehension.

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

HOL Constant

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

$$\begin{aligned}
&\forall \alpha\ \gamma \bullet\ ScPrec\ \alpha\ \gamma \Leftrightarrow \\
&\quad (\{\alpha;\ \gamma\} \subseteq Syntax \wedge \gamma = MkComp\ \alpha) \\
&\vee\ (\exists \beta \bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkEq\ (\alpha,\ \beta)) \\
&\vee\ (\exists \beta \bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkEq\ (\beta,\ \alpha)) \\
&\vee\ (\exists \beta \bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkMem\ (\alpha,\ \beta)) \\
&\vee\ (\exists \beta \bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkMem\ (\beta,\ \alpha)) \\
&\vee\ (\exists ord\ fs \bullet\ \alpha \in_g fs \wedge \{\alpha;\ \gamma\} \subseteq Syntax \wedge \gamma = MkTf\ (ord,\ fs))
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{ScPrec_tc_\in_thm} = \\
&\quad \vdash \forall\ x\ y \bullet\ ScPrec\ x\ y \Rightarrow tc\ \$\in_g\ x\ y
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{well_founded_ScPrec_thm} = \\
&\quad \vdash well_founded\ ScPrec
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{well_founded_tcScPrec_thm} = \\
&\quad \vdash well_founded\ (tc\ ScPrec)
\end{aligned}$$

SML

$$\begin{aligned}
&val\ \boldsymbol{SC_INDUCTION_T} = WFCV_INDUCTION_T\ well_founded_ScPrec_thm; \\
&val\ \boldsymbol{sc_induction_tac} = wfcv_induction_tac\ well_founded_ScPrec_thm;
\end{aligned}$$

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.

HOL Constant

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

$$WfTerms = Terms\ Syntax$$

HOL Constant

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

$$WfComp = \{x \mid x \in WfTerms \wedge IsComp\ x\}$$

HOL Constant

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

$$WfForms = Formulas\ Syntax$$

syntax_disj_thm =

$\vdash \forall\ x$

$\bullet\ x \in Syntax$

$\Rightarrow (\exists\ ord \bullet\ ordinal\ ord \wedge x = MkVar\ ord)$

$\vee (\exists\ f \bullet\ f \in WfForms \wedge x = MkComp\ f)$

$\vee (\exists\ t1\ t2 \bullet\ t1 \in WfTerms \wedge t2 \in WfTerms \wedge x = MkEq\ (t1, t2))$

$\vee (\exists\ t1\ t2 \bullet\ t1 \in WfTerms \wedge t2 \in WfTerms \wedge x = MkMem\ (t1, t2))$

$\vee (\exists\ ord\ fs \bullet\ ordinal\ ord \wedge X_g\ fs \subseteq WfForms \wedge x = MkTf\ (ord, fs))$

is_fc_clauses =

$\vdash \forall\ x$

$\bullet\ x \in Syntax$

$\Rightarrow (IsVar\ x \Rightarrow (\exists\ ord \bullet\ ordinal\ ord \wedge x = MkVar\ ord))$

$\wedge (IsComp\ x \Rightarrow (\exists\ f \bullet\ f \in WfForms \wedge x = MkComp\ f))$

$\wedge (IsEq\ x$

$\Rightarrow (\exists\ t1\ t2 \bullet\ t1 \in WfTerms \wedge t2 \in WfTerms \wedge x = MkEq\ (t1, t2)))$

$\wedge (IsMem\ x$

$\Rightarrow (\exists\ t1\ t2 \bullet\ t1 \in WfTerms \wedge t2 \in WfTerms \wedge x = MkMem\ (t1, t2)))$

$\wedge (IsTf\ x$

$\Rightarrow (\exists\ ord\ fs$

$\bullet\ ordinal\ ord \wedge X_g\ fs \subseteq WfForms \wedge x = MkTf\ (ord, fs)))$

syn_proj_clauses =

$\vdash (\forall\ ord \bullet\ VarNum\ (MkVar\ ord) = ord)$

$\wedge (\forall\ f \bullet\ CompBody\ (MkComp\ f) = f)$

$\wedge (\forall\ l\ r \bullet\ AtomLhs\ (MkEq\ (l, r)) = l)$

$\wedge (\forall\ l\ r \bullet\ AtomRhs\ (MkEq\ (l, r)) = r)$

$\wedge (\forall\ l\ r \bullet\ AtomLhs\ (MkMem\ (l, r)) = l)$

$\wedge (\forall\ l\ r \bullet\ AtomRhs\ (MkMem\ (l, r)) = r)$

$\wedge (\forall\ v\ f \bullet\ TfVars\ (MkTf\ (v, f)) = v)$

$\wedge (\forall\ v\ f \bullet\ TfForms\ (MkTf\ (v, f)) = f)$

is_fc_clauses2 =

$\vdash \forall\ x$

$\bullet\ x \in Syntax$

$\Rightarrow (IsVar\ x \Rightarrow ordinal\ (VarNum\ x))$

$\wedge (IsComp\ x \Rightarrow CompBody\ x \in WfForms)$

$\wedge (IsEq\ x \Rightarrow AtomLhs\ x \in WfTerms \wedge AtomRhs\ x \in WfTerms)$

$\wedge (IsMem\ x \Rightarrow AtomLhs\ x \in WfTerms \wedge AtomRhs\ x \in WfTerms)$

$\wedge (IsTf\ x$

$\Rightarrow ordinal\ (TfVars\ x) \wedge (\forall\ y \bullet\ y \in_g TfForms\ x \Rightarrow y \in WfForms))$

syn_comp_fc_clauses =

$\vdash (\forall\ x \bullet\ MkVar\ x \in Syntax \Rightarrow ordinal\ x)$

$\wedge (\forall\ x \bullet\ MkComp\ x \in Syntax \Rightarrow x \in WfForms)$

$\wedge$ ($\forall$ l $r\bullet$ $MkEq$ $(l, r) \in Syntax \Rightarrow l \in WfTerms \wedge r \in WfTerms$)
$\wedge$ ($\forall$ l $r\bullet$ $MkMem$ $(l, r) \in Syntax \Rightarrow l \in WfTerms \wedge r \in WfTerms$)
$\wedge$ ($\forall$ v f
$\bullet$ $MkTf$ $(v, f) \in Syntax \Rightarrow ordinal\ v \wedge (\forall\ y\bullet\ y \in_g f \Rightarrow y \in WfForms)) = TEX$

ft_syntax_thm =
$\vdash \forall\ x\bullet\ x \in WfForms \vee x \in WfTerms \Rightarrow x \in Syntax$

formula_cases_thm =
$\vdash \forall\ x\bullet\ x \in WfForms \vee IsForm\ x \Rightarrow IsEq\ x \vee IsMem\ x \vee IsTf\ x$

term_cases_thm =
$\vdash \forall\ x\bullet\ x \in WfTerms \vee IsTerm\ x \Rightarrow IsVar\ x \vee IsComp\ x$

scprec_fc_clauses =
$\vdash \forall\ \alpha\ \beta\ \gamma\ ord\ fs$
$\bullet\ \gamma \in Syntax$
$\Rightarrow \gamma = MkComp\ \alpha$
$\vee \gamma = MkEq\ (\alpha, \beta)$
$\vee \gamma = MkEq\ (\beta, \alpha)$
$\vee \gamma = MkMem\ (\alpha, \beta)$
$\vee \gamma = MkMem\ (\beta, \alpha)$
$\vee \gamma = MkTf\ (ord, fs) \wedge \alpha \in_g fs$
$\Rightarrow ScPrec\ \alpha\ \gamma$

scprec_fc_clauses2 =
$\vdash \forall\ t$
$\bullet\ t \in Syntax$
$\Rightarrow (IsComp\ t \Rightarrow ScPrec\ (CompBody\ t)\ t)$
$\wedge$ ($IsEq\ t \vee IsMem\ t \Rightarrow ScPrec\ (AtomLhs\ t)\ t \wedge ScPrec\ (AtomRhs\ t)\ t$)
$\wedge$ ($IsTf\ t \Rightarrow (\forall\ f\bullet\ f \in_g TfForms\ t \Rightarrow ScPrec\ f\ t)$)

3.4 Proof Contexts

SML

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

force_new_pc "ICsyn";
merge_pcs ["hol", "'GS1", "'ICsyn"] "ICsyn";
commit_pc "ICsyn";

force_new_pc "ICsyn1";
merge_pcs ["hol1", "'GS1", "'ICsyn"] "ICsyn1";
commit_pc "ICsyn1";
```

4 SEMANTICS

SML

```
open_theory "ICsyn";
force_new_theory "ICsem";
force_new_pc "'ICsem";
merge_pcs ["'savedthm_cs_∃_proof"] "'ICsem";
(* set_merge_pcs ["hol", "'GS1", "'ICsyn", "'ICsem"]; *)
set_merge_pcs ["ICsyn", "'ICsem"];
```

The semantics will be defined as a functor which transforms partial membership and equality relations, and is parameterised by a domain set (which gives the range of the quantifiers).

We want to be able to evaluate membership claims between closed comprehensions. This is done by substituting the comprehension on the left for the variable zero in the body of the comprehension on the right and evaluating the result. Alternatively, evaluating the body of the comprehension on the right in a context which consists just of the assignment to variable zero of the comprehsion on the left. When we reach atomic equations and membership claims during this evaluation we look them up using the initial values for the equality and membership relations. We can only do this with closed comprehensions, so at this point, if not before we must substitute values from the context for the free variables in the comprehensions (these variables will be bound by quantifiers so we will be doing this substitution for every comprehension in turn).

So we need to be able to evaluate and to instantiate, evaluation taking place down to the level of atomic formulae and instantiation below that level (note that the terms in an atomic formulae will usually be comprehesions and hence may contain non-atomic formulae).

The following type abbreviations are intended to make the specification more readable:

EV Expression value

VA Variable assignment

TD Term denotation

TV Truth value

FD Formula denotation

R Partial relation

PR Pair of partial relations

SML

```
declare_type_abbrev("EV", [], ⌜:GS⌝);
declare_type_abbrev("VA", [], ⌜:GS × (GS → EV)⌝);
declare_type_abbrev("TD", [], ⌜:VA → EV⌝);
declare_type_abbrev("TV", [], ⌜:BOOL + ONE⌝);
declare_type_abbrev("FD", [], ⌜:VA → TV⌝);
declare_type_abbrev("R", [], ⌜:GS → GS → TV⌝);
declare_type_abbrev("PR", [], ⌜:R × R⌝);
```

Sometimes in the semantics similar operations are defined several times over different kinds of objects. In these cases we sometimes distinguish between the variants by using the same name subscripted by and indicator of the type.

$name_r$ type ⌜:R⌝

$name_{pr}$ type ⌜:PR⌝

4.1 Substitution

This will be an inductive definition over the relevant syntactic structures, but the definition is given in pieces.

To define the semantics of comprehension we need to be able to modify a variable assignment. The modification required is to insert a new value for variable zero shifting all the existing values up one variable number.

HOL Constant

$$\boldsymbol{VaRan} : VA \to EV\ SET$$

$$\forall va\bullet\ VaRan\ va = \{tv \mid \exists\alpha\bullet\ \alpha <_o Fst\ va \wedge tv = Snd\ va\ \alpha\}$$

HOL Constant

$$\boldsymbol{InsertVar} : VA \to EV \to VA$$

$$\forall d\ f\ tv\bullet\ InsertVar\ (d, f)\ tv =$$
$$(suc_o\ d, \lambda\beta\bullet\ if\ \beta = \varnothing_g\ then\ tv\ else\ f\ (\beta --_o (Nat_g\ 1)))$$

This one concatenates two variable assignments.

HOL Constant

$$\boldsymbol{InsertVars} : VA \to VA \to VA$$

$$\forall\alpha\ \beta\ f1\ f2\bullet\ InsertVars\ (\alpha, f1)\ (\beta, f2) =$$
$$(\alpha +_o \beta, \lambda\gamma\bullet\ if\ \gamma <_o \alpha\ then\ f1\ \gamma\ else\ f2\ (\gamma --_o \alpha))$$

We now define the operation of substituting into a term or formula values for free variables as specified by an assignment. The substitution is for free variables and the operator requires as an argument the number of bound variables for any particular context (the numbers of the free variables a shifted upwards as bound variables are introduced.

HOL Constant

$$\boldsymbol{SubstTerm} : ((GS \times VA) \times GS \to GS) \to ((GS \times VA) \times GS) \to GS$$

$$SubstTerm = \lambda sf\ ((\alpha, (\beta, f)), t)\bullet$$
$$\quad if\ t \in WfTerms$$
$$\quad then\ \ if\ IsVar\ t$$
$$\qquad then\ if\ \alpha \leq_o VarNum\ t \wedge VarNum\ t <_o (\alpha +_o \beta)$$
$$\qquad\quad then\ f\ ((VarNum\ t) --_o \alpha)$$
$$\qquad\quad else\ t$$
$$\qquad else\ sf\ ((\alpha, (InsertVar\ (\beta, f)\ \varnothing_g)), CompBody\ t)$$
$$\quad else\ t$$

The following gives us a well-founded ordering on the parameters to SubstForm which will be useful in proving its well-definedness.

HOL Constant

$$\boldsymbol{SubOrder} : ((GS \times VA) \times GS) \to ((GS \times VA) \times GS) \to BOOL$$

$$SubOrder = \lambda s\ t \bullet tc\ ScPrec\ (Snd\ s)\ (Snd\ t)$$

The following results will be useful later in proving the existence of a fixed point.

First some points about the constituents of terms.

$$\boldsymbol{tran\_suborder\_thm} =$$
$$\vdash trans\ SubOrder$$

$$\boldsymbol{tran\_suborder\_thm2} =$$
$$\vdash \forall\ s\ t\ u \bullet SubOrder\ s\ t \wedge SubOrder\ t\ u \Rightarrow SubOrder\ s\ u$$

$$\boldsymbol{substterm\_lemma1} =$$
$$\vdash \forall\ sf1\ sf2\ t$$
$$\bullet\ (\forall\ \alpha\ \beta\_f\ y$$
$$\bullet\ SubOrder\ ((\alpha,\ \beta\_f),\ y)\ ((\alpha,\ \beta\_f),\ t)$$
$$\Rightarrow sf1\ ((\alpha,\ \beta\_f),\ y) = sf2\ ((\alpha,\ \beta\_f),\ y))$$
$$\Rightarrow (\forall\ \alpha\ \beta\_f \bullet SubstTerm\ sf1\ ((\alpha,\ \beta\_f),\ t) = SubstTerm\ sf2\ ((\alpha,\ \beta\_f),\ t))$$

$$\boldsymbol{well\_founded\_SubOrder\_thm} =$$
$$\vdash well\_founded\ SubOrder$$

Substitution over formulae is defined by recursion over the syntax and is thefore as the fixed point of a functor.

HOL Constant

$$\boldsymbol{SubstAtom} : ((GS \times VA) \times GS \to GS) \to ((GS \times VA) \times GS \to GS)$$

$$SubstAtom = \lambda sf\ ((\alpha,\ va),\ f) \bullet$$
$$(if\ IsEq\ f\ then\ MkEq\ else\ MkMem)$$
$$(SubstTerm\ sf\ ((\alpha,\ va),\ (AtomLhs\ f)),$$
$$SubstTerm\ sf\ ((\alpha,\ va),\ (AtomRhs\ f)))$$

$$substatom\_lemma1 =$$
$$\vdash \forall\ sf1\ sf2\ f$$
$$\bullet\ Snd\ f \in Syntax$$
$$\Rightarrow IsEq\ (Snd\ f) \vee IsMem\ (Snd\ f)$$
$$\Rightarrow (\forall\ y \bullet SubOrder\ y\ f \Rightarrow sf1\ y = sf2\ y)$$
$$\Rightarrow SubstAtom\ sf1\ f = SubstAtom\ sf2\ f$$

HOL Constant

$$\textbf{\textit{SubstTf}} : ((GS \times VA) \times GS \to GS) \to ((GS \times VA) \times GS \to GS)$$

$$SubstTf = \lambda sf\ ((\alpha, va), f)\bullet$$
$$\quad let\ \nu = TfVars\ f$$
$$\quad and\ fs = TfForms\ f$$
$$\quad in\ (MkTf\ (\nu, (Imagep\ (\lambda f\bullet\ sf\ ((\nu +_o \alpha, va), f))\ fs)))$$

$$\textbf{\textit{substtf\_lemma1}} =$$
$$\vdash \forall\ sf1\ sf2\ f$$
$$\bullet\ Snd\ f \in Syntax$$
$$\Rightarrow IsTf\ (Snd\ f)$$
$$\Rightarrow (\forall\ y\bullet\ SubOrder\ y\ f \Rightarrow sf1\ y = sf2\ y)$$
$$\Rightarrow SubstTf\ sf1\ f = SubstTf\ sf2\ f$$

HOL Constant

$$\textbf{\textit{SubstFormFunct}} : ((GS \times VA) \times GS \to GS) \to ((GS \times VA) \times GS \to GS)$$

$$SubstFormFunct = \lambda sf\ ((\alpha, va), f)\bullet$$
$$\quad if\ f \in Syntax$$
$$\quad then\ if\ IsEq\ f \vee IsMem\ f$$
$$\qquad then\ SubstAtom\ sf\ ((\alpha, va), f)$$
$$\qquad else\ if\ IsTf\ f$$
$$\qquad\quad then\ SubstTf\ sf\ ((\alpha, va), f)$$
$$\qquad\quad else\ \varnothing_g$$
$$\quad else\ \varnothing_g$$

$$\textbf{\textit{substformfunct\_lemma1}} =$$
$$\vdash SubstFormFunct\ respects\ SubOrder$$

HOL Constant

$$\textbf{\textit{SubstForm}} : (GS \times VA) \times GS \to GS$$

$$SubstForm = fix\ SubstFormFunct$$

$$\textbf{\textit{substformfunct\_fixp\_lemma}} =$$
$$\vdash SubstForm = SubstFormFunct\ SubstForm$$

$$\textbf{\textit{substformfunct\_thm}} =$$
$$\vdash SubstForm$$
$$= (\lambda\ ((\alpha, va), f)$$
$$\bullet\ if\ f \in Syntax$$
$$then$$

```
        if IsEq f ∨ IsMem f
        then SubstAtom SubstForm ((α, va), f)
        else if IsTf f
        then SubstTf SubstForm ((α, va), f)
        else ∅_g
      else ∅_g)

substformfunct_thm2 =
  ⊢ ∀ x
    • SubstForm x
      = (if Snd x ∈ Syntax
        then
          if IsEq (Snd x) ∨ IsMem (Snd x)
          then SubstAtom SubstForm x
          else if IsTf (Snd x)
          then SubstTf SubstForm x
          else ∅_g
        else ∅_g)
```

4.2 Evaluation

Now we define the evaluation of formulae. The definition assumes that partial relations for the atomic formulae are available, and is the main part of the definition of a functor which transforms membership and equality relations. Interpretations of non-well-founded set theories are then sought as fixed points of this functor over subsets of the infinitary comprehensions.

Partial relations are represented by curried functions with values of type ⌜:BOOL + ONE⌝, a type abbreviated as ⌜:TV⌝. To improve readability of the specification the three truth values are give names as follows:

HOL Constant

```
pTrue : TV
──────────────────────────────────────
pTrue = InL T
```

HOL Constant

```
pFalse : TV
──────────────────────────────────────
pFalse = InL F
```

HOL Constant

```
pU : TV
──────────────────────────────────────
pU = InR One
```

$$
\begin{array}{l}
\boldsymbol{tv\_cases\_thm} = \\
\quad \vdash \forall\, x\bullet\ x = pTrue \vee x = pFalse \vee x = pU
\end{array}
$$

$$
\begin{array}{l}
\boldsymbol{tv\_distinct\_clauses} = \\
\quad \vdash \neg\ pTrue = pFalse \\
\qquad \wedge \neg\ pTrue = pU \\
\qquad \wedge \neg\ pFalse = pTrue \\
\qquad \wedge \neg\ pFalse = pU \\
\qquad \wedge \neg\ pU = pTrue \\
\qquad \wedge \neg\ pU = pFalse
\end{array}
$$

SML

$$
\begin{array}{l}
declare\_infix\ (300,\ \texttt{"}\in_{\boldsymbol{v}}\texttt{"}); \\
declare\_infix\ (300,\ \texttt{"}=_{\boldsymbol{v}}\texttt{"});
\end{array}
$$

Now we begin the specification of evaluation with the evaluation of atomic formulae.

HOL Constant

$$
\begin{array}{l}
\boldsymbol{EvalAtom} : PR \to (VA \times GS) \to TV \\
\hline
\forall(\$\in_v{:}R)\ (\$=_v{:}R)\bullet\ EvalAtom\ (\$\in_v,\ \$=_v) = (\lambda(va,\ f)\bullet \\
\quad (if\ IsEq\ f\ then\ \$=_v\ else\ \$\in_v) \\
\qquad (SubstTerm\ SubstForm\ ((\varnothing_g,\ va),\ (AtomLhs\ f))) \\
\qquad (SubstTerm\ SubstForm\ ((\varnothing_g,\ va),\ (AtomRhs\ f))))
\end{array}
$$

It is helpful for the subsequent proofs to structure the specification of *EvalTf*, pulling out the following definition, which shows the set of truth values arising from the quantification occuring in this construct.

HOL Constant

$$
\begin{array}{l}
\boldsymbol{EvalTf\_results} : GS\ SET \to ((VA \times GS) \to TV) \to (VA \times GS) \to (TV)\ SET \\
\hline
\forall V\bullet\ EvalTf\_results\ V = \lambda ef\ (va,\ f)\bullet \\
\quad let\ \nu = TfVars\ f \\
\quad and\ fs = TfForms\ f \\
\quad in\ \{pb\ | \\
\qquad \exists v\ f\bullet\ VaRan\ (\nu,\ v) \subseteq V \\
\qquad \wedge\ f \in_g fs \\
\qquad \wedge\ pb = ef\ (InsertVars\ (\nu,\ v)\ va,\ f)\}
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\boldsymbol{EvalTf\_tf} : TV\ SET \to TV \\
\hline
\forall results\bullet\ EvalTf\_tf\ results = \\
\quad if\ results \subseteq \{pTrue\}\ then\ pFalse \\
\qquad else\ if\ (pFalse) \in results\ then\ pTrue \\
\qquad else\ pU
\end{array}
$$

HOL Constant

$$\textbf{\textit{EvalTf}} : (GS\ SET \times R \times R) \to ((VA \times GS) \to TV) \to (VA \times GS) \to TV$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet\ EvalTf\ (V, \$\in_v, \$=_v) = \lambda ef\ (va, f)\bullet$$
$$EvalTf\_tf\ (EvalTf\_results\ V\ ef\ (va, f))$$

HOL Constant

$$\textbf{\textit{SubOrder2}} : (VA \times GS) \to (VA \times GS) \to BOOL$$

$$SubOrder2 = \lambda s\ t\bullet\ tc\ ScPrec\ (Snd\ s)\ (Snd\ t)$$

tran_suborder2_thm =

$$\vdash trans\ SubOrder2$$

tran_suborder2_thm2 =

$$\vdash \forall\ s\ t\ u\bullet\ SubOrder2\ s\ t \wedge SubOrder2\ t\ u \Rightarrow SubOrder2\ s\ u$$

well_founded_SubOrder2_thm =

$$\vdash well\_founded\ SubOrder2$$

evaltf_lemma1 =

$$\vdash \forall\ (v, m, e)\ ef1\ ef2\ vaf$$
$$\bullet\ Snd\ vaf \in Syntax$$
$$\Rightarrow IsTf\ (Snd\ vaf)$$
$$\Rightarrow (\forall\ y\bullet\ SubOrder2\ y\ vaf \Rightarrow ef1\ y = ef2\ y)$$
$$\Rightarrow EvalTf\ (v, m, e)\ ef1\ vaf = EvalTf\ (v, m, e)\ ef2\ vaf$$

HOL Constant

$$\textbf{\textit{EvalFormFunct}} : (GS\ SET \times R \times R) \to (VA \times GS \to TV)$$
$$\to (VA \times GS \to TV)$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet\ EvalFormFunct\ (V, \$\in_v, \$=_v) = (\lambda ef\ (va, f)\bullet$$
$$if\ f \in WfForms$$
$$then$$
$$if\ IsEq\ f \vee IsMem\ f$$
$$then\ EvalAtom\ (\$\in_v, \$=_v)\ (va, f)$$
$$else\ EvalTf\ (V, \$\in_v, \$=_v)\ ef\ (va, f)$$
$$else\ pU)$$

HOL Constant

$$\boldsymbol{EvalForm} : (GS\ SET \times R \times R) \to VA \times GS \to TV$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet\ EvalForm\ (V, \$\in_v, \$=_v) = fix\ (EvalFormFunct\ (V, \$\in_v, \$=_v))$$

$$\boldsymbol{evalformfunct\_fixp\_lemma} =$$
$$\vdash \forall\ (V, \$\in_v, \$=_v)$$
$$\bullet\ EvalForm\ (V, \$\in_v, \$=_v)$$
$$= EvalFormFunct\ (V, \$\in_v, \$=_v)\ (EvalForm\ (V, \$\in_v, \$=_v))$$

$$\boldsymbol{evalformfunct\_thm} =$$
$$\vdash \forall\ (V, \$\in_v, \$=_v)$$
$$\bullet\ EvalForm\ (V, \$\in_v, \$=_v)$$
$$= (\lambda\ (va, f)$$
$$\bullet\ if\ f \in WfForms$$
$$then$$
$$if\ IsEq\ f \vee IsMem\ f$$
$$then\ EvalAtom\ (\$\in_v, \$=_v)\ (va, f)$$
$$else\ EvalTf\ (V, \$\in_v, \$=_v)\ (EvalForm\ (V, \$\in_v, \$=_v))\ (va, f)$$
$$else\ pU)$$

$$\boldsymbol{evalformfunct\_thm2} =$$
$$\vdash \forall\ (V, \$\in_v, \$=_v)\ x$$
$$\bullet\ EvalForm\ (V, \$\in_v, \$=_v)\ x$$
$$= (if\ Snd\ x \in WfForms$$
$$then$$
$$if\ IsEq\ (Snd\ x) \vee IsMem\ (Snd\ x)$$
$$then\ EvalAtom\ (\$\in_v, \$=_v)\ x$$
$$else\ EvalTf\ (V, \$\in_v, \$=_v)\ (EvalForm\ (V, \$\in_v, \$=_v))\ x$$
$$else\ pU)$$

4.3 Membership and Equality

Note that in the evaluation of formulae above *EvalForm* atomic membership and equality relations are evaluated by reference to given membership and equality relationships.

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

HOL Constant

$$\boldsymbol{MemRel} : (GS\ SET \times R \times R) \to R$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet$$
$$MemRel\ (V, \$\in_v, \$=_v) = \lambda e\ s\bullet$$
$$let\ va = (Nat_g\ 1, \lambda v\bullet\ e)$$
$$in\ EvalForm\ (V, \$\in_v, \$=_v)\ (va, (snd\ s))$$

The equality of two closed comprehensions is determined here by evaluating the formula in which the bodies of the two comprehensions are first combined into an equivalence and then universally quantified over the common variable.

HOL Constant

$$\boldsymbol{EqRel} : (GS\ SET \times R \times R) \to R$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet\ EqRel\ (V, \$\in_v, \$=_v) = \lambda l\ r\bullet$$
$$if\ l \in V \wedge r \in V$$
$$then$$
$$if\ \exists c\bullet\ c \in V \wedge ((c \in_v l) = pU \vee (c \in_v r) = pU)$$
$$then\ pU$$
$$else\ if\ ((\lambda c\bullet\ c \in V \wedge \neg\ (c \in_v l) = pU) = (\lambda c\bullet\ c \in V \wedge \neg\ (c \in_v r) = pU))$$
$$then\ pTrue\ else\ pFalse$$
$$else\ pU$$

The semantics of equality and membership are to be combined into a parameterised functor of which we seek fixed points. These fixed points yield interpretations of non-well-founded set theories (or as a special and easy test case an image of our well-founded set theory).

4.4 The Semantic Functor

The required functor is a monotonic functor over equality and membership relation pairs, parameterised by the domain of discourse V.

HOL Constant

$$\boldsymbol{SemanticFunctor} : GS\ SET \to PR \to PR$$

$$\forall V\ (\$\in_v{:}R)\ (\$=_v{:}R)\bullet$$
$$SemanticFunctor\ V = \lambda(\$\in_v, \$=_v)\bullet\ (MemRel\ (V, \$\in_v, \$=_v), EqRel\ (V, \$\in_v, \$=_v))$$

Minor variations on this are possible, such as obtaining the equality relation from the new rather than the old membership relation, or even iterating the construction of the equality to an extensional

fixed point in each application of the semantic functor. However, these devices are unlikely to affect the fixed points and their effect on the complexity of proofs might as well be negative as positive.

5 THE EXISTENCE OF FIXED POINTS

When I began working with infinitary comprehension the idea was to use it for interpretations of the usual first order language of set theory. This was to have been realised by identifying large subsets V of *WfComp* for which there exists a fixed point of the functor *SemanticFunctor V* in which the equality and membership relations are total over V. It is easy to believe that there are such subsets which include copies of the well-founded sets we started out with, and also, for example, all the PolySets and many other useful non-well-founded sets.

Before reaching the stage at which reasoning about such fixed points could be undertaken it occurred to me that for my intended application, the classical set theories might well be dispensed with, possibly resulting in significant savings.

The intended application was to languages suitable for the formal computer assisted development of mathematics and its applications, and the next stage in the construction of such languages was to be an illative lambda calculus. An illative lambda calculus must be effectively a many valued logic, for there is no type system or other means which prevents the consideration of arbitrary lambda terms as propositions. For application in this context, the effort of coming up with a two-valued membership relation may not be beneficial. There must still be a system of type assignment which allows the user to work with subdomains better behaved than the whole ontology, and if this works as well as it needs to work, then the presence of sets which are not really sets in the classical sense will not be problematic, and effort directed toward their elimination may prove not to be beneficial.

I therefore propose at least initially to explore the option of going straight from partial fixed points (meaning in this case, fixed points, over the whole of *WfComp*, which are partial membership and equality relations) to an illative lambda calculus whose domain of discourse is a partition of the whole of *WfComp*.

At this stage it is not clear whether any old fixed point will do or whether we have to be more fussy than that, so I will start with an arbitrary fixed point and see how far I get. To do this I do have to prove that there does exist a fixed point, and this is to be realised by demonstrating that the semantic functor is monotone and therefore has a least fixed point.

5.1 Monotonicity

To establish the existence of fixed points it is helpful to show that the semantic functor is monotonic.

It is therefore necessary to define the ordering relative to which it is monotonic.

SML
```
declare_infix(300, "≤t");
declare_infix(300, "≤ts");
declare_infix(300, "≤r");
declare_infix(300, "≤pr");
```

First an ordering on the “truth values” is defined.

HOL Constant

$\$\leq_{\boldsymbol{t}} : (TV) \to (TV) \to BOOL$

$\forall\ t1\ t2 \bullet$
$\quad t1 \leq_t t2 \Leftrightarrow t1 = t2 \vee t1 = pU$

$\boldsymbol{\leq_{t}\text{-}refl\_thm} =$
$\quad \vdash \forall\ x \bullet\ x \leq_t x$

$\boldsymbol{\leq_{t}\text{-}trans\_thm} =$
$\quad \vdash \forall\ x\ y\ z \bullet\ x \leq_t y \wedge y \leq_t z \Rightarrow x \leq_t z$

$\boldsymbol{\leq_{t}\text{-}clauses} =$
$\vdash pU \leq_t pTrue$
$\quad \wedge\ pU \leq_t pFalse$
$\quad \wedge\ \neg\ pTrue \leq_t pU$
$\quad \wedge\ \neg\ pFalse \leq_t pU$
$\quad \wedge\ \neg\ pFalse \leq_t pTrue$
$\quad \wedge\ \neg\ pTrue \leq_t pFalse$

Then an ordering on partial relations over term values.

HOL Constant

$\$\leq_{\boldsymbol{r}} : R \to R \to BOOL$

$\forall\ r1\ r2 \bullet$
$\quad r1 \leq_r r2 \Leftrightarrow \forall x\ y \bullet\ (r1\ x\ y) \leq_t (r2\ x\ y)$

and an ordering over pairs of partial relations (membership and equality partial relations).

HOL Constant

$\$\leq_{\boldsymbol{pr}} : PR \to PR \to BOOL$

$\forall\ r12\ s12 \bullet$
$\quad r12 \leq_{pr} s12 \Leftrightarrow (Fst\ r12) \leq_r (Fst\ s12) \wedge (Snd\ r12) \leq_r (Snd\ s12)$

Our aim is to prove the monotonicity of the semantic functor for every domain which is a subset of the well formed comprehensions. The relevant notion of monotonicity is defined here.

HOL Constant

$\boldsymbol{MonoFunct} : (PR \to PR) \to BOOL$

$\forall\ f \bullet\ \ MonoFunct\ f \Leftrightarrow \forall v\ w\ x\ y \bullet\ (v,w) \leq_{pr} (x,y) \Rightarrow (f\ (v,w)) \leq_{pr} (f\ (x,y))$

In order to prove monotonicity of the semantic functor various lemmas about the functions used in defining the semantic functor are needed, often expressing monotonicity of objects of various types.

5.1.1 EvalAtom

The required characteristic of
it EvalAtom is straightforward to define and prove.

The following is the property of partial truth predicates parameterised by pairs of partial relations of being monotonic with respect to those relations.

HOL Constant

$$\boldsymbol{MonoPprF} : (PR \to 'a \to TV) \to BOOL$$

$$\forall f \bullet\ MonoPprF\ f \Leftrightarrow \forall\ v\ w\ x\ y \bullet$$
$$(v,\ w) \leq_{pr} (x,\ y) \Rightarrow \forall z{:}'a \bullet\ f\ (v,\ w)\ z \leq_t f\ (x,\ y)\ z$$

$$\boldsymbol{evalatom\_monotone\_lemma} =$$
$$\vdash \forall\ v\ w\ x\ y$$
$$\bullet\ \$\leq_{pr}\ (v,\ w)\ (x,\ y) \Rightarrow (\forall\ z \bullet\ EvalAtom\ (v,\ w)\ z \leq_t EvalAtom\ (x,\ y)\ z)$$

5.1.2 Monotonicity of Membership and Equality

The following are lemmas specific to the definitions of *EqRel* and *MemRel*.

$$\boldsymbol{eqrel\_mono\_thm} =$$
$$\vdash \forall\ V\ v\ w\ x\ y$$
$$\bullet\ \$\leq_{pr}\ (x,\ y)\ (v,\ w) \Rightarrow \$\leq_r\ (EqRel\ (V,\ x,\ y))\ (EqRel\ (V,\ v,\ w))$$

$$\boldsymbol{monpprf\_memrel\_lemma1} =$$
$$\vdash \forall\ V$$
$$\bullet\ MonoPprF\ (\lambda\ (m,\ e) \bullet\ EvalForm\ (V,\ m,\ e))$$
$$\Rightarrow MonoPprF\ (\lambda\ (m,\ e)\ (l,\ r) \bullet\ MemRel\ (V,\ m,\ e)\ l\ r)$$

5.1.3 Monotonicity of EvalTf

This result is more difficult. Because *EvalTf* participates in the recursion over the syntactic structure of comprehensions which is used to define evaluation of formulae, it is supplied with an evaluation function on whose behaviour its monotonicity depends.

The definition of *EvalTf* has been split into three parts to break up the proof. The first part obtains a set of "truth values" (true, false or unknown) and we here define an ordering over these sets.

HOL Constant

$$\boldsymbol{\$\leq_{ts}} : (TV\ SET) \to (TV\ SET) \to BOOL$$

$$\forall\ m\ n \bullet$$
$$m \leq_{ts} n \Leftrightarrow$$
$$(\forall x \bullet\ x \in m \Rightarrow \exists y \bullet\ y \in n \wedge x \leq_t y)$$
$$\wedge\quad (\forall y \bullet\ y \in n \Rightarrow \exists x \bullet\ x \in m \wedge x \leq_t y)$$

$\leq_{ts}\_clauses =$
$\vdash \forall\ s\ t$
$\bullet\ \$\leq_{ts}\ s\ t$
$\Rightarrow\ pTrue \in s$
$\Rightarrow\ pTrue \in t \wedge pFalse \in s$
$\Rightarrow\ pFalse \in t \wedge pU \in t$
$\Rightarrow\ pU \in s \wedge (s = \{\} \Leftrightarrow t = \{\})$

$mono\_evaltf\_\leq_{ts}\_lemma1 =$
$\vdash \forall\ s\ t \bullet\ \$\leq_{ts}\ s\ t \Rightarrow s \subseteq \{pTrue\} \Rightarrow s \subseteq \{pTrue\}$

$mono\_evaltftf\_lemma =$
$\vdash \forall\ res1\ res2 \bullet\ \$\leq_{ts}\ res1\ res2 \Rightarrow EvalTf\_tf\ res1 \leq_t EvalTf\_tf\ res2$

5.1.4 The Semantic Functor

$mono\_semanticfunct\_lemma1 =$
$\vdash \forall\ V$
$\bullet\ MonoPprF\ (\lambda\ (m,\ e) \bullet\ EvalForm\ (V,\ m,\ e))$
$\Rightarrow\ MonoFunct\ (SemanticFunctor\ V)$

$mono\_semanticfunctor\_thm =$
$\vdash \forall\ V \bullet\ MonoFunct\ (SemanticFunctor\ V)$

5.2 The Least Partial Fixed Point

Having established the monotonicity of the semantic functor we can obtain a fixed point for any class V. This will not immediately yield an interpretation of first order set theory because the fixed point will be a pair of relations which are not total over V.

To obtain interpretations of first order set theory we must chose V so as to omit sets which are problematic, and we may view this as seeking a notion of consistency appropriate to this context, i.e. a notion of when a property is consistently reifiable.

The partial fixed points may however be independently useful in applications where the systems of interest are not first order set theories, as in our case where an illative lamdba calculus is sought.

In order to obtain a least fixed point we must define the greatest lower bound of a set of pairs of relations.

For this purpose we need to know the greatest lower bound of a set of truth values.

HOL Constant

$glb_{ts} : TV\ SET \rightarrow TV$

$\forall\ tvs \bullet\ glb_{ts}\ tvs =$
$\quad if\ tvs = \{pTrue\}\ then\ pTrue$
$\quad else\ if\ tvs = \{pFalse\}\ then\ pFalse$
$\quad else\ pU$

We need to know that this is indeed the greatest lower bound, and to express this claim we define the relevan notion of lower bound.

HOL Constant

$$\boldsymbol{IsLb_{ts}} : TV\ SET \to TV \to BOOL$$

$$\forall\ tvs\ lb\bullet\ IsLb_{ts}\ tvs\ lb =$$
$$\quad \forall tv\bullet\ tv \in tvs \Rightarrow lb \leq_t tv$$

$$\boldsymbol{glb_{ts}\_thm} =$$
$$\vdash \forall\ tvs$$
$$\bullet\ (\exists\ tv\bullet\ tv \in tvs)$$
$$\Rightarrow\ IsLb_{ts}\ tvs\ (glb_{ts}\ tvs)$$
$$\wedge\ (\forall\ tv\bullet\ IsLb_{ts}\ tvs\ tv \Rightarrow tv \leq_t glb_{ts}\ tvs)$$

The greatest lower bound of a set of partial relations is:

HOL Constant

$$\boldsymbol{glb_{rs}} : R\ SET \to R$$

$$\forall\ spr\bullet\ glb_{rs}\ spr =$$
$$\quad \lambda x\ y\bullet\ glb_{ts}\ \{tv \mid \exists pr\bullet\ pr \in spr \wedge pr\ x\ y = tv\}$$

The relevant notion of lower bound is:

HOL Constant

$$\boldsymbol{IsLb_{rs}} : R\ SET \to R \to BOOL$$

$$\forall\ prs\ lb\bullet\ IsLb_{rs}\ prs\ lb =$$
$$\quad \forall pr\bullet\ pr \in prs \Rightarrow lb \leq_r pr$$

The greatest lower bound of a set of pairs of partial relations is then:

HOL Constant

$$\boldsymbol{glb_{spr}} : PR\ SET \to PR$$

$$\forall\ sppr\bullet\ glb_{spr}\ sppr =$$
$$\quad (glb_{rs}\ \{pr \mid \exists ppr\bullet\ ppr \in sppr \wedge pr = Fst\ ppr\},$$
$$\quad\ glb_{rs}\ \{pr \mid \exists ppr\bullet\ ppr \in sppr \wedge pr = Snd\ ppr\})$$

and the least fixed point of a functor over pairs of partial relations is:

HOL Constant

$$\boldsymbol{lfp_{pr}} : (PR \to PR) \to PR$$

$$\forall\ f\bullet\ lfp_{pr}\ f =$$
$$\quad glb_{spr}\ \{ppr \mid \$\leq_{pr}\ (f\ ppr)\ ppr\}$$

5.3 Alternative Definition of Least Fixed Point

The proof that the above definition does yield a least fixed point is eluding me, so I thought I would try harder to make use of the fixed point result already available

6 SEMANTICS

Now meaning is attached to the representatives. This is done in such a way as to yield a functor from one membership structure to another, of which we will then be seeking useful partial fixed points.

This functor will be compounded from maps for individual constructs which, parameterised by the incoming structure, return the extension of the interpreted construct in that context. These extensions are collected to give both a new membership relation and an equivalence relative to which that relation is extensional.

The membership relations concerned are over equivalance classes of representatives, and the domains of the structures are partial partitions of the set of representations.

The semantics is therefore defined in a piecemeal way for each constructor in turn and then sewn together to give the required functor.

The fact that we have inter-defined membership and equality relationships, and also that we are expecting at best partial fixed points, together hint that we might benefit from working in the more general context of "Boolean Values Models" (see, for example, [1]), so I propose to begin in that more general context and see how it goes. i.e. the equality and membership relations will be functions yielding values in an arbitrary boolean algebra rather than classical relationships, and the relationship between them will be as prescribed for boolean valued models unless a problem is found with this.

7 The Theory ICsyn

7.1 Parents

$U\_orders \quad GS$

7.2 Children

$ICsem$

7.3 Constants

$\boldsymbol{MkVar}$	$GS \to GS$
$\boldsymbol{IsVar}$	$GS \to BOOL$
$\boldsymbol{VarNum}$	$GS \to GS$
$\boldsymbol{MkComp}$	$GS \to GS$
$\boldsymbol{IsComp}$	$GS \to BOOL$
$\boldsymbol{CompBody}$	$GS \to GS$
$\boldsymbol{MkEq}$	$GS \times GS \to GS$
$\boldsymbol{IsEq}$	$GS \to BOOL$
$\boldsymbol{MkMem}$	$GS \times GS \to GS$
$\boldsymbol{IsMem}$	$GS \to BOOL$
$\boldsymbol{AtomLhs}$	$GS \to GS$
$\boldsymbol{AtomRhs}$	$GS \to GS$
$\boldsymbol{MkTf}$	$GS \times GS \to GS$
$\boldsymbol{IsTf}$	$GS \to BOOL$
$\boldsymbol{TfVars}$	$GS \to GS$
$\boldsymbol{TfForms}$	$GS \to GS$
$\boldsymbol{MkNot}$	$GS \to GS$
$\boldsymbol{IsTerm}$	$GS \to BOOL$
$\boldsymbol{Terms}$	$GS\ \mathbb{P} \to GS\ \mathbb{P}$
$\boldsymbol{IsForm}$	$GS \to BOOL$
$\boldsymbol{Formulas}$	$GS\ \mathbb{P} \to GS\ \mathbb{P}$
$\boldsymbol{RepClosed}$	$GS\ \mathbb{P} \to BOOL$
$\boldsymbol{Syntax}$	$GS\ \mathbb{P}$
$\boldsymbol{ScPrec}$	$GS \to GS \to BOOL$
$\boldsymbol{WfTerms}$	$GS\ \mathbb{P}$
$\boldsymbol{WfComp}$	$GS\ \mathbb{P}$
$\boldsymbol{WfForms}$	$GS\ \mathbb{P}$

7.4 Definitions

$\boldsymbol{MkVar}$	$\vdash \forall\ v \bullet\ MkVar\ v = Nat_g\ 0 \mapsto_g v$
$\boldsymbol{IsVar}$	$\vdash \forall\ t \bullet\ IsVar\ t \Leftrightarrow fst\ t = Nat_g\ 0$
$\boldsymbol{VarNum}$	$\vdash\ VarNum = snd$
$\boldsymbol{MkComp}$	$\vdash \forall\ f \bullet\ MkComp\ f = Nat_g\ 1 \mapsto_g f$
$\boldsymbol{IsComp}$	$\vdash \forall\ t \bullet\ IsComp\ t \Leftrightarrow fst\ t = Nat_g\ 1$
$\boldsymbol{CompBody}$	$\vdash\ CompBody = snd$
$\boldsymbol{MkEq}$	$\vdash \forall\ lr \bullet\ MkEq\ lr = Nat_g\ 2 \mapsto_g Fst\ lr \mapsto_g Snd\ lr$
$\boldsymbol{IsEq}$	$\vdash \forall\ t \bullet\ IsEq\ t \Leftrightarrow fst\ t = Nat_g\ 2$

MkMem $\vdash \forall\ lr\bullet\ MkMem\ lr = Nat_g\ 3 \mapsto_g Fst\ lr \mapsto_g Snd\ lr$

IsMem $\vdash \forall\ t\bullet\ IsMem\ t \Leftrightarrow fst\ t = Nat_g\ 3$

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

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

MkTf $\vdash \forall\ vc\bullet\ MkTf\ vc = Nat_g\ 4 \mapsto_g Fst\ vc \mapsto_g Snd\ vc$

IsTf $\vdash \forall\ t\bullet\ IsTf\ t \Leftrightarrow fst\ t = Nat_g\ 4$

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

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

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

IsTerm $\vdash \forall\ t\bullet\ IsTerm\ t \Leftrightarrow IsVar\ t \vee IsComp\ t$

Terms $\vdash \forall\ s\bullet\ Terms\ s = \{x | x \in s \wedge IsTerm\ x\}$

IsForm $\vdash \forall\ x\bullet\ IsForm\ x \Leftrightarrow IsMem\ x \vee IsEq\ x \vee IsTf\ x$

Formulas $\vdash \forall\ s\bullet\ Formulas\ s = \{x | x \in s \wedge IsForm\ x\}$

RepClosed $\vdash \forall\ s$
$\bullet\ RepClosed\ s$
$\Leftrightarrow (\forall\ ord\bullet\ ordinal\ ord \Rightarrow MkVar\ ord \in s)$
$\wedge (\forall\ f\bullet\ f \in Formulas\ s \Rightarrow MkComp\ f \in s)$
$\wedge (\forall\ t1\ t2$
$\bullet\ t1 \in Terms\ s \wedge t2 \in Terms\ s$
$\Rightarrow MkEq\ (t1,\ t2) \in s \wedge MkMem\ (t1,\ t2) \in s)$
$\wedge (\forall\ ord\ fs$
$\bullet\ ordinal\ ord \wedge X_g\ fs \subseteq Formulas\ s$
$\Rightarrow MkTf\ (ord,\ fs) \in s)$

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

ScPrec $\vdash \forall\ \alpha\ \gamma$
$\bullet\ ScPrec\ \alpha\ \gamma$
$\Leftrightarrow \{\alpha;\ \gamma\} \subseteq Syntax \wedge \gamma = MkComp\ \alpha$
$\vee (\exists\ \beta\bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkEq\ (\alpha,\ \beta))$
$\vee (\exists\ \beta\bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkEq\ (\beta,\ \alpha))$
$\vee (\exists\ \beta\bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkMem\ (\alpha,\ \beta))$
$\vee (\exists\ \beta\bullet\ \{\alpha;\ \beta;\ \gamma\} \subseteq Syntax \wedge \gamma = MkMem\ (\beta,\ \alpha))$
$\vee (\exists\ ord\ fs$
$\bullet\ \alpha \in_g fs$
$\wedge \{\alpha;\ \gamma\} \subseteq Syntax$
$\wedge \gamma = MkTf\ (ord,\ fs))$

WfTerms $\vdash WfTerms = Terms\ Syntax$

WfComp $\vdash WfComp = \{x | x \in WfTerms \wedge IsComp\ x\}$

WfForms $\vdash WfForms = Formulas\ Syntax$

7.5 Theorems

tc_TranClsr_thm
$\vdash \forall\ r\bullet\ (Universe,\ tc\ r) = TranClsr\ (Universe,\ r)$

Is_not_fc_clauses
$\vdash (\forall\ x$
$\bullet\ IsVar\ x$
$\Rightarrow \neg\ IsComp\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsMem\ x \wedge \neg\ IsTf\ x)$
$\wedge (\forall\ x$
$\bullet\ IsComp\ x$
$\Rightarrow \neg\ IsVar\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsMem\ x \wedge \neg\ IsTf\ x)$
$\wedge (\forall\ x$

$$
\begin{array}{l}
\quad \bullet\ IsEq\ x \\
\quad\quad \Rightarrow \neg\ IsComp\ x \\
\quad\quad\quad \wedge \neg\ IsVar\ x \\
\quad\quad\quad \wedge \neg\ IsMem\ x \\
\quad\quad\quad \wedge \neg\ IsTf\ x) \\
\quad \wedge (\forall\ x \\
\quad \bullet\ IsMem\ x \\
\quad\quad \Rightarrow \neg\ IsComp\ x \wedge \neg\ IsVar\ x \wedge \neg\ IsEq\ x \wedge \neg\ IsTf\ x) \\
\quad \wedge (\forall\ x \\
\quad \bullet\ IsTf\ x \\
\quad\quad \Rightarrow \neg\ IsComp\ x \\
\quad\quad\quad \wedge \neg\ IsVar\ x \\
\quad\quad\quad \wedge \neg\ IsEq\ x \\
\quad\quad\quad \wedge \neg\ IsMem\ x)
\end{array}
$$

terms_mono_thm

$$
\vdash \forall\ s\ t \bullet\ s \subseteq t \Rightarrow Terms\ s \subseteq Terms\ t
$$

formulas_mono_thm

$$
\vdash \forall\ s\ t \bullet\ s \subseteq t \Rightarrow Formulas\ s \subseteq Formulas\ t
$$

repclosed_syntax_thm

$$
\begin{array}{l}
\vdash (\forall\ ord \bullet\ ordinal\ ord \Rightarrow MkVar\ ord \in Syntax) \\
\quad \wedge (\forall\ f \bullet\ f \in Formulas\ Syntax \Rightarrow MkComp\ f \in Syntax) \\
\quad \wedge (\forall\ t1\ t2 \\
\quad \bullet\ t1 \in Terms\ Syntax \wedge t2 \in Terms\ Syntax \\
\quad\quad \Rightarrow MkEq\ (t1, t2) \in Syntax \\
\quad\quad\quad \wedge MkMem\ (t1, t2) \in Syntax) \\
\quad \wedge (\forall\ ord\ fs \\
\quad \bullet\ ordinal\ ord \\
\quad\quad\quad \wedge (\forall\ x \bullet\ x \in X_g\ fs \Rightarrow x \in Formulas\ Syntax) \\
\quad\quad \Rightarrow MkTf\ (ord, fs) \in Syntax)
\end{array}
$$

repclosed_syntax_lemma1

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

repclosed_syntax_lemma2

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

term_or_formula_thm

$$
\vdash \forall\ x \bullet\ x \in Syntax \Rightarrow IsTerm\ x \vee IsForm\ x
$$

well_founded_ScPrec_thm

$$
\vdash well\_founded\ ScPrec
$$

well_founded_tcScPrec_thm

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

syntax_disj_thm

$$
\begin{array}{l}
\vdash \forall\ x \\
\quad \bullet\ x \in Syntax \\
\quad\quad \Rightarrow (\exists\ ord \bullet\ ordinal\ ord \wedge x = MkVar\ ord) \\
\quad\quad\quad \vee (\exists\ f \bullet\ f \in WfForms \wedge x = MkComp\ f) \\
\quad\quad\quad \vee (\exists\ t1\ t2 \\
\quad\quad\quad \bullet\ t1 \in WfTerms \\
\quad\quad\quad\quad \wedge\ t2 \in WfTerms \\
\quad\quad\quad\quad \wedge\ x = MkEq\ (t1, t2)) \\
\quad\quad\quad \vee (\exists\ t1\ t2 \\
\quad\quad\quad \bullet\ t1 \in WfTerms \\
\quad\quad\quad\quad \wedge\ t2 \in WfTerms
\end{array}
$$

$$\wedge\ x = MkMem\ (t1,\ t2))$$
$$\vee\ (\exists\ ord\ fs$$
$$\bullet\ ordinal\ ord$$
$$\wedge\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in WfForms)$$
$$\wedge\ x = MkTf\ (ord,\ fs))$$

is_fc_clauses

$$\vdash \forall\ x$$
$$\bullet\ x \in Syntax$$
$$\Rightarrow\ (IsVar\ x$$
$$\Rightarrow\ (\exists\ ord\bullet\ ordinal\ ord \wedge x = MkVar\ ord))$$
$$\wedge\ (IsComp\ x$$
$$\Rightarrow\ (\exists\ f\bullet\ f \in WfForms \wedge x = MkComp\ f))$$
$$\wedge\ (IsEq\ x$$
$$\Rightarrow\ (\exists\ t1\ t2$$
$$\bullet\ t1 \in WfTerms$$
$$\wedge\ t2 \in WfTerms$$
$$\wedge\ x = MkEq\ (t1,\ t2)))$$
$$\wedge\ (IsMem\ x$$
$$\Rightarrow\ (\exists\ t1\ t2$$
$$\bullet\ t1 \in WfTerms$$
$$\wedge\ t2 \in WfTerms$$
$$\wedge\ x = MkMem\ (t1,\ t2)))$$
$$\wedge\ (IsTf\ x$$
$$\Rightarrow\ (\exists\ ord\ fs$$
$$\bullet\ ordinal\ ord$$
$$\wedge\ (\forall\ y\bullet\ y \in_g fs \Rightarrow y \in WfForms)$$
$$\wedge\ x = MkTf\ (ord,\ fs)))$$

syn_proj_clauses

$$\vdash (\forall\ ord\bullet\ VarNum\ (MkVar\ ord) = ord)$$
$$\wedge\ (\forall\ f\bullet\ CompBody\ (MkComp\ f) = f)$$
$$\wedge\ (\forall\ l\ r\bullet\ AtomLhs\ (MkEq\ (l,\ r)) = l)$$
$$\wedge\ (\forall\ l\ r\bullet\ AtomRhs\ (MkEq\ (l,\ r)) = r)$$
$$\wedge\ (\forall\ l\ r\bullet\ AtomLhs\ (MkMem\ (l,\ r)) = l)$$
$$\wedge\ (\forall\ l\ r\bullet\ AtomRhs\ (MkMem\ (l,\ r)) = r)$$
$$\wedge\ (\forall\ v\ f\bullet\ TfVars\ (MkTf\ (v,\ f)) = v)$$
$$\wedge\ (\forall\ v\ f\bullet\ TfForms\ (MkTf\ (v,\ f)) = f)$$

is_fc_clauses2

$$\vdash \forall\ x$$
$$\bullet\ x \in Syntax$$
$$\Rightarrow\ (IsVar\ x \Rightarrow ordinal\ (VarNum\ x))$$
$$\wedge\ (IsComp\ x \Rightarrow CompBody\ x \in WfForms)$$
$$\wedge\ (IsEq\ x$$
$$\Rightarrow\ AtomLhs\ x \in WfTerms \wedge AtomRhs\ x \in WfTerms)$$
$$\wedge\ (IsMem\ x$$
$$\Rightarrow\ AtomLhs\ x \in WfTerms \wedge AtomRhs\ x \in WfTerms)$$
$$\wedge\ (IsTf\ x$$
$$\Rightarrow\ ordinal\ (TfVars\ x)$$
$$\wedge\ (\forall\ y\bullet\ y \in_g TfForms\ x \Rightarrow y \in WfForms))$$

syn_con_neq_clauses

$$\vdash \forall\ v\ f\ lr\ vf$$
$$\bullet\ \neg\ MkVar\ v = MkComp\ f$$

$$
\begin{array}{l}
\quad \wedge \neg\ MkVar\ v = MkEq\ lr \\
\quad \wedge \neg\ MkVar\ v = MkMem\ lr \\
\quad \wedge \neg\ MkVar\ v = MkTf\ vf \\
\quad \wedge \neg\ MkComp\ f = MkVar\ v \\
\quad \wedge \neg\ MkComp\ f = MkEq\ lr \\
\quad \wedge \neg\ MkComp\ f = MkMem\ lr \\
\quad \wedge \neg\ MkComp\ f = MkTf\ vf \\
\quad \wedge \neg\ MkEq\ lr = MkVar\ v \\
\quad \wedge \neg\ MkEq\ lr = MkComp\ f \\
\quad \wedge \neg\ MkEq\ lr = MkMem\ lr \\
\quad \wedge \neg\ MkEq\ lr = MkTf\ vf \\
\quad \wedge \neg\ MkMem\ lr = MkVar\ v \\
\quad \wedge \neg\ MkMem\ lr = MkComp\ f \\
\quad \wedge \neg\ MkMem\ lr = MkEq\ lr \\
\quad \wedge \neg\ MkMem\ lr = MkTf\ vf \\
\quad \wedge \neg\ MkTf\ vf = MkVar\ v \\
\quad \wedge \neg\ MkTf\ vf = MkComp\ f \\
\quad \wedge \neg\ MkTf\ vf = MkEq\ lr \\
\quad \wedge \neg\ MkTf\ vf = MkMem\ lr
\end{array}
$$

syn_comp_fc_clauses

$$
\begin{array}{l}
\vdash (\forall\ x\bullet\ MkVar\ x \in Syntax \Rightarrow ordinal\ x) \\
\quad \wedge (\forall\ x\bullet\ MkComp\ x \in Syntax \Rightarrow x \in WfForms) \\
\quad \wedge (\forall\ l\ r \\
\quad \bullet\ MkEq\ (l,\ r) \in Syntax \Rightarrow l \in WfTerms \wedge r \in WfTerms) \\
\quad \wedge (\forall\ l\ r \\
\quad \bullet\ MkMem\ (l,\ r) \in Syntax \\
\qquad \Rightarrow l \in WfTerms \wedge r \in WfTerms) \\
\quad \wedge (\forall\ v\ f \\
\quad \bullet\ MkTf\ (v,\ f) \in Syntax \\
\qquad \Rightarrow ordinal\ v \wedge (\forall\ y\bullet\ y \in_g f \Rightarrow y \in WfForms))
\end{array}
$$

ft_syntax_thm

$$\vdash \forall\ x\bullet\ x \in WfForms \vee x \in WfTerms \Rightarrow x \in Syntax$$

formula_cases_thm

$$
\begin{array}{l}
\vdash \forall\ x \\
\quad \bullet\ x \in WfForms \vee IsForm\ x \Rightarrow IsEq\ x \vee IsMem\ x \vee IsTf\ x
\end{array}
$$

term_cases_thm

$$\vdash \forall\ x\bullet\ x \in WfTerms \vee IsTerm\ x \Rightarrow IsVar\ x \vee IsComp\ x$$

scprec_fc_clauses

$$
\begin{array}{l}
\vdash \forall\ \alpha\ \beta\ \gamma\ ord\ fs \\
\quad \bullet\ \gamma \in Syntax \\
\qquad \Rightarrow \gamma = MkComp\ \alpha \\
\qquad\quad \vee \gamma = MkEq\ (\alpha,\ \beta) \\
\qquad\quad \vee \gamma = MkEq\ (\beta,\ \alpha) \\
\qquad\quad \vee \gamma = MkMem\ (\alpha,\ \beta) \\
\qquad\quad \vee \gamma = MkMem\ (\beta,\ \alpha) \\
\qquad\quad \vee \gamma = MkTf\ (ord,\ fs) \wedge \alpha \in_g fs \\
\qquad \Rightarrow ScPrec\ \alpha\ \gamma
\end{array}
$$

scprec_fc_clauses2

$$
\begin{array}{l}
\vdash \forall\ t \\
\quad \bullet\ t \in Syntax \\
\qquad \Rightarrow (IsComp\ t \Rightarrow ScPrec\ (CompBody\ t)\ t)
\end{array}
$$

$$
\begin{array}{l}
\wedge\ (\mathit{IsEq}\ t \vee \mathit{IsMem}\ t \\
\quad \Rightarrow \mathit{ScPrec}\ (\mathit{AtomLhs}\ t)\ t \\
\qquad \wedge\ \mathit{ScPrec}\ (\mathit{AtomRhs}\ t)\ t) \\
\wedge\ (\mathit{IsTf}\ t \\
\quad \Rightarrow (\forall\ f \bullet f \in_g \mathit{TfForms}\ t \Rightarrow \mathit{ScPrec}\ f\ t))
\end{array}
$$

8 The Theory ICsem

8.1 Parents

ICsyn

8.2 Constants

$\boldsymbol{VaRan}$	$VA \to EV\ \mathbb{P}$
$\boldsymbol{InsertVar}$	$VA \to EV \to VA$
$\boldsymbol{InsertVars}$	$VA \to VA \to VA$
$\boldsymbol{SubstTerm}$	$((EV \times VA) \times EV \to EV) \to (EV \times VA) \times EV \to EV$
$\boldsymbol{SubOrder}$	$(EV \times VA) \times EV \to (EV \times VA) \times EV \to BOOL$
$\boldsymbol{SubstAtom}$	$((EV \times VA) \times EV \to EV) \to (EV \times VA) \times EV \to EV$
$\boldsymbol{SubstTf}$	$((EV \times VA) \times EV \to EV) \to (EV \times VA) \times EV \to EV$
$\boldsymbol{SubstFormFunct}$	$((EV \times VA) \times EV \to EV) \to (EV \times VA) \times EV \to EV$
$\boldsymbol{SubstForm}$	$(EV \times VA) \times EV \to EV$
$\boldsymbol{pTrue}$	TV
$\boldsymbol{pFalse}$	TV
$\boldsymbol{pU}$	TV
$\boldsymbol{EvalAtom}$	$PR \to VA \times EV \to TV$
$\boldsymbol{EvalTf\_results}$	$EV\ \mathbb{P} \to (VA \times EV \to TV) \to VA \times EV \to TV\ \mathbb{P}$
$\boldsymbol{EvalTf\_tf}$	$TV\ \mathbb{P} \to TV$
$\boldsymbol{EvalTf}$	$EV\ \mathbb{P} \times PR \to (VA \times EV \to TV) \to VA \times EV \to TV$
$\boldsymbol{SubOrder2}$	$VA \times EV \to VA \times EV \to BOOL$
$\boldsymbol{EvalFormFunct}$	$EV\ \mathbb{P} \times PR \to (VA \times EV \to TV) \to VA \times EV \to TV$
$\boldsymbol{EvalForm}$	$EV\ \mathbb{P} \times PR \to VA \times EV \to TV$
$\boldsymbol{MemRel}$	$EV\ \mathbb{P} \times PR \to R$
$\boldsymbol{EqRel}$	$EV\ \mathbb{P} \times PR \to R$
$\boldsymbol{SemanticFunctor}$	$EV\ \mathbb{P} \to PR \to PR$
$\$\leq_t$	$TV \to TV \to BOOL$
$\$\leq_r$	$R \to R \to BOOL$
$\$\leq_{pr}$	$PR \to PR \to BOOL$
$\boldsymbol{MonoFunct}$	$(PR \to PR) \to BOOL$
$\boldsymbol{MonoPprF}$	$(PR \to 'a \to TV) \to BOOL$
$\$\leq_{ts}$	$TV\ \mathbb{P} \to TV\ \mathbb{P} \to BOOL$
$\boldsymbol{glb_{ts}}$	$TV\ \mathbb{P} \to TV$
$\boldsymbol{IsLb_{ts}}$	$TV\ \mathbb{P} \to TV \to BOOL$
$\boldsymbol{glb_{rs}}$	$R\ \mathbb{P} \to R$
$\boldsymbol{IsLb_{rs}}$	$R\ \mathbb{P} \to R \to BOOL$
$\boldsymbol{glb_{spr}}$	$R \leftrightarrow R \to PR$
$\boldsymbol{lfp_{pr}}$	$(PR \to PR) \to PR$

8.3 Type Abbreviations

$\boldsymbol{EV}$	EV
$\boldsymbol{VA}$	VA
$\boldsymbol{TD}$	TD
$\boldsymbol{TV}$	TV
$\boldsymbol{FD}$	FD
$\boldsymbol{R}$	R
$\boldsymbol{PR}$	PR

8.4 Fixity

Right Infix 300:

$$=_{\boldsymbol{v}} \quad \in_{\boldsymbol{v}} \quad \leq_{\boldsymbol{pr}} \quad \leq_{\boldsymbol{r}} \quad \leq_{\boldsymbol{t}} \quad \leq_{\boldsymbol{ts}}$$

8.5 Definitions

$VaRan$ $\vdash \forall\ va$
- $VaRan\ va = \{tv | \exists\ \alpha \bullet\ \alpha <_o Fst\ va \wedge tv = Snd\ va\ \alpha\}$

$InsertVar$ $\vdash \forall\ d\ f\ tv$
- $InsertVar\ (d, f)\ tv$
  $= (suc_o\ d,$
  $(\lambda\ \beta$
  - $if\ \beta = \varnothing_g$
    $then\ tv$
    $else\ f\ (\beta --_o Nat_g\ 1)))$

$InsertVars$ $\vdash \forall\ \alpha\ \beta\ f1\ f2$
- $InsertVars\ (\alpha, f1)\ (\beta, f2)$
  $= (\alpha +_o \beta,$
  $(\lambda\ \gamma \bullet\ if\ \gamma <_o \alpha\ then\ f1\ \gamma\ else\ f2\ (\gamma --_o \alpha)))$

$SubstTerm$ $\vdash SubstTerm$
$= (\lambda\ sf\ ((\alpha, \beta, f), t)$
- $if\ t \in WfTerms$
  $then$
  $if\ IsVar\ t$
  $then$
  $if\ \alpha \leq_o VarNum\ t \wedge VarNum\ t <_o \alpha +_o \beta$
  $then\ f\ (VarNum\ t --_o \alpha)$
  $else\ t$
  $else\ sf\ ((\alpha, InsertVar\ (\beta, f)\ \varnothing_g), CompBody\ t)$
  $else\ t)$

$SubOrder$ $\vdash SubOrder = (\lambda\ s\ t \bullet\ tc\ ScPrec\ (Snd\ s)\ (Snd\ t))$

$SubstAtom$ $\vdash SubstAtom$
$= (\lambda\ sf\ ((\alpha, va), f)$
- $(if\ IsEq\ f\ then\ MkEq\ else\ MkMem)$
  $(SubstTerm\ sf\ ((\alpha, va), AtomLhs\ f),$
  $SubstTerm\ sf\ ((\alpha, va), AtomRhs\ f)))$

$SubstTf$ $\vdash SubstTf$
$= (\lambda\ sf\ ((\alpha, va), f)$
- $(let\ \nu = TfVars\ f\ and\ fs = TfForms\ f$
  $in\ MkTf$

$(\nu,\ Imagep\ (\lambda\ f\bullet\ sf\ ((\nu\ +_o\ \alpha,\ va),\ f))\ fs)))$

SubstFormFunct

$\vdash\ SubstFormFunct$
$= (\lambda\ sf\ ((\alpha,\ va),\ f)$
$\bullet\ if\ f \in Syntax$
$then$
$if\ IsEq\ f \vee IsMem\ f$
$then\ SubstAtom\ sf\ ((\alpha,\ va),\ f)$
$else\ if\ IsTf\ f$
$then\ SubstTf\ sf\ ((\alpha,\ va),\ f)$
$else\ \varnothing_g$
$else\ \varnothing_g)$

SubstForm $\vdash\ SubstForm = fix\ SubstFormFunct$

pTrue $\vdash\ pTrue = InL\ T$

pFalse $\vdash\ pFalse = InL\ F$

pU $\vdash\ pU = InR\ One$

EvalAtom $\vdash \forall\ \$\in_v\ \$=_v$
$\bullet\ EvalAtom\ (\$\in_v,\ \$=_v)$
$= (\lambda\ (va,\ f)$
$\bullet\ (if\ IsEq\ f\ then\ \$=_v\ else\ \$\in_v)$
$(SubstTerm\ SubstForm\ ((\varnothing_g,\ va),\ AtomLhs\ f))$
$(SubstTerm\ SubstForm\ ((\varnothing_g,\ va),\ AtomRhs\ f)))$

EvalTf_results

$\vdash \forall\ V$
$\bullet\ EvalTf\_results\ V$
$= (\lambda\ ef\ (va,\ f)$
$\bullet\ (let\ \nu = TfVars\ f\ and\ fs = TfForms\ f$
$in\ \{pb$
$|\exists\ v\ f$
$\bullet\ VaRan\ (\nu,\ v) \subseteq V$
$\wedge\ f \in_g fs$
$\wedge\ pb = ef\ (InsertVars\ (\nu,\ v)\ va,\ f)\}))$

EvalTf_tf $\vdash \forall\ results$
$\bullet\ EvalTf\_tf\ results$
$= (if\ results \subseteq \{pTrue\}$
$then\ pFalse$
$else\ if\ pFalse \in results$
$then\ pTrue$
$else\ pU)$

EvalTf $\vdash \forall\ V\ \$\in_v\ \$=_v$
$\bullet\ EvalTf\ (V,\ \$\in_v,\ \$=_v)$
$= (\lambda\ ef\ (va,\ f)$
$\bullet\ EvalTf\_tf\ (EvalTf\_results\ V\ ef\ (va,\ f)))$

SubOrder2 $\vdash\ SubOrder2 = (\lambda\ s\ t\bullet\ tc\ ScPrec\ (Snd\ s)\ (Snd\ t))$

EvalFormFunct

$\vdash \forall\ V\ \$\in_v\ \$=_v$
$\bullet\ EvalFormFunct\ (V,\ \$\in_v,\ \$=_v)$
$= (\lambda\ ef\ (va,\ f)$
$\bullet\ if\ f \in WfForms$
$then$
$if\ IsEq\ f \vee IsMem\ f$

$$
\begin{array}{ll}
 & \quad \mathit{then}\ \mathit{EvalAtom}\ (\$\in_v, \$=_v)\ (va, f) \\
 & \quad \mathit{else}\ \mathit{EvalTf}\ (V, \$\in_v, \$=_v)\ ef\ (va, f) \\
 & \mathit{else}\ pU) \\
\boldsymbol{EvalForm} & \vdash \forall\ V\ \$\in_v\ \$=_v \\
 & \bullet\ \mathit{EvalForm}\ (V, \$\in_v, \$=_v) \\
 & \quad = \mathit{fix}\ (\mathit{EvalFormFunct}\ (V, \$\in_v, \$=_v)) \\
\boldsymbol{MemRel} & \vdash \forall\ V\ \$\in_v\ \$=_v \\
 & \bullet\ \mathit{MemRel}\ (V, \$\in_v, \$=_v) \\
 & \quad = (\lambda\ e\ s \\
 & \quad \bullet\ (\mathit{let}\ va = (\mathit{Nat}_g\ 1, (\lambda\ v\bullet\ e)) \\
 & \quad\quad \mathit{in}\ \mathit{EvalForm}\ (V, \$\in_v, \$=_v)\ (va, \mathit{snd}\ s))) \\
\boldsymbol{EqRel} & \vdash \forall\ V\ \$\in_v\ \$=_v \\
 & \bullet\ \mathit{EqRel}\ (V, \$\in_v, \$=_v) \\
 & \quad = (\lambda\ l\ r \\
 & \quad \bullet\ \mathit{if}\ l \in V \wedge r \in V \\
 & \quad\quad \mathit{then} \\
 & \quad\quad\quad \mathit{if}\ \exists\ c\bullet\ c \in V \wedge (c \in_v l = pU \vee c \in_v r = pU) \\
 & \quad\quad\quad \mathit{then}\ pU \\
 & \quad\quad\quad \mathit{else\ if} \\
 & \quad\quad\quad\quad (\lambda\ c\bullet\ c \in V \wedge \neg\ c \in_v l = pU) \\
 & \quad\quad\quad\quad\quad = (\lambda\ c\bullet\ c \in V \wedge \neg\ c \in_v r = pU) \\
 & \quad\quad\quad \mathit{then}\ pTrue \\
 & \quad\quad\quad \mathit{else}\ pFalse \\
 & \quad\quad \mathit{else}\ pU) \\
\boldsymbol{SemanticFunctor} & \\
 & \vdash \forall\ V\ \$\in_v\ \$=_v \\
 & \bullet\ \mathit{SemanticFunctor}\ V \\
 & \quad = (\lambda\ (\$\in_v, \$=_v) \\
 & \quad \bullet\ (\mathit{MemRel}\ (V, \$\in_v, \$=_v), \\
 & \quad\quad \mathit{EqRel}\ (V, \$\in_v, \$=_v))) \\
\boldsymbol{\leq_t} & \vdash \forall\ t1\ t2\bullet\ t1 \leq_t t2 \Leftrightarrow t1 = t2 \vee t1 = pU \\
\boldsymbol{\leq_r} & \vdash \forall\ r1\ r2\bullet\ r1 \leq_r r2 \Leftrightarrow (\forall\ x\ y\bullet\ r1\ x\ y \leq_t r2\ x\ y) \\
\boldsymbol{\leq_{pr}} & \vdash \forall\ r12\ s12 \\
 & \bullet\ r12 \leq_{pr} s12 \\
 & \quad \Leftrightarrow \mathit{Fst}\ r12 \leq_r \mathit{Fst}\ s12 \wedge \mathit{Snd}\ r12 \leq_r \mathit{Snd}\ s12 \\
\boldsymbol{MonoFunct} & \vdash \forall\ f \\
 & \bullet\ \mathit{MonoFunct}\ f \\
 & \quad \Leftrightarrow (\forall\ v\ w\ x\ y \\
 & \quad \bullet\ (v, w) \leq_{pr} (x, y) \Rightarrow f\ (v, w) \leq_{pr} f\ (x, y)) \\
\boldsymbol{MonoPprF} & \vdash \forall\ f \\
 & \bullet\ \mathit{MonoPprF}\ f \\
 & \quad \Leftrightarrow (\forall\ v\ w\ x\ y \\
 & \quad \bullet\ (v, w) \leq_{pr} (x, y) \\
 & \quad\quad \Rightarrow (\forall\ z\bullet\ f\ (v, w)\ z \leq_t f\ (x, y)\ z)) \\
\boldsymbol{\leq_{ts}} & \vdash \forall\ m\ n \\
 & \bullet\ m \leq_{ts} n \\
 & \quad \Leftrightarrow (\forall\ x\bullet\ x \in m \Rightarrow (\exists\ y\bullet\ y \in n \wedge x \leq_t y)) \\
 & \quad\quad \wedge (\forall\ y\bullet\ y \in n \Rightarrow (\exists\ x\bullet\ x \in m \wedge x \leq_t y)) \\
\boldsymbol{glb_{ts}} & \vdash \forall\ tvs \\
 & \bullet\ glb_{ts}\ tvs \\
 & \quad = (\mathit{if}\ tvs = \{pTrue\}
\end{array}
$$

$\quad\quad then\ pTrue$
$\quad\quad else\ if\ tvs = \{pFalse\}$
$\quad\quad then\ pFalse$
$\quad\quad else\ pU)$

$\boldsymbol{IsLb_{ts}}$ $\vdash \forall\ tvs\ lb$
$\bullet\ IsLb_{ts}\ tvs\ lb \Leftrightarrow (\forall\ tv\bullet\ tv \in tvs \Rightarrow lb \leq_t tv)$

$\boldsymbol{glb_{rs}}$ $\vdash \forall\ spr$
$\bullet\ glb_{rs}\ spr$
$= (\lambda\ x\ y$
$\bullet\ glb_{ts}\ \{tv | \exists\ pr\bullet\ pr \in spr \land pr\ x\ y = tv\})$

$\boldsymbol{IsLb_{rs}}$ $\vdash \forall\ prs\ lb$
$\bullet\ IsLb_{rs}\ prs\ lb \Leftrightarrow (\forall\ pr\bullet\ pr \in prs \Rightarrow lb \leq_r pr)$

$\boldsymbol{glb_{spr}}$ $\vdash \forall\ sppr$
$\bullet\ glb_{spr}\ sppr$
$= (glb_{rs}\ \{pr | \exists\ ppr\bullet\ ppr \in sppr \land pr = Fst\ ppr\},$
$glb_{rs}\ \{pr | \exists\ ppr\bullet\ ppr \in sppr \land pr = Snd\ ppr\})$

$\boldsymbol{lfp_{pr}}$ $\vdash \forall\ f\bullet\ lfp_{pr}\ f = glb_{spr}\ \{ppr | f\ ppr \leq_{pr} ppr\}$

8.6 Theorems

tran_suborder_thm
$\vdash trans\ SubOrder$

tran_suborder_thm2
$\vdash \forall\ s\ t\ u\bullet\ SubOrder\ s\ t \land SubOrder\ t\ u \Rightarrow SubOrder\ s\ u$

well_founded_SubOrder_thm
$\vdash well\_founded\ SubOrder$

substterm_lemma1
$\vdash \forall\ sf1\ sf2\ p$
$\bullet\ (\forall\ y\bullet\ SubOrder\ y\ p \Rightarrow sf1\ y = sf2\ y)$
$\Rightarrow SubstTerm\ sf1\ p = SubstTerm\ sf2\ p$

substatom_lemma1
$\vdash \forall\ sf1\ sf2\ f$
$\bullet\ Snd\ f \in Syntax$
$\Rightarrow IsEq\ (Snd\ f) \lor IsMem\ (Snd\ f)$
$\Rightarrow (\forall\ y\bullet\ SubOrder\ y\ f \Rightarrow sf1\ y = sf2\ y)$
$\Rightarrow SubstAtom\ sf1\ f = SubstAtom\ sf2\ f$

substtf_lemma1
$\vdash \forall\ sf1\ sf2\ f$
$\bullet\ Snd\ f \in Syntax$
$\Rightarrow IsTf\ (Snd\ f)$
$\Rightarrow (\forall\ y\bullet\ SubOrder\ y\ f \Rightarrow sf1\ y = sf2\ y)$
$\Rightarrow SubstTf\ sf1\ f = SubstTf\ sf2\ f$

substformfunct_lemma1
$\vdash SubstFormFunct\ respects\ SubOrder$

substformfunct_fixp_lemma
$\vdash SubstForm = SubstFormFunct\ SubstForm$

substformfunct_thm
$\vdash SubstForm$
$= (\lambda\ ((\alpha,\ va),\ f)$
$\bullet\ if\ f \in Syntax$
$then$

$if\ IsEq\ f \lor IsMem\ f$
$then\ SubstAtom\ SubstForm\ ((\alpha,\ va),\ f)$
$else\ if\ IsTf\ f$
$then\ SubstTf\ SubstForm\ ((\alpha,\ va),\ f)$
$else\ \varnothing_g$
$else\ \varnothing_g)$

substformfunct_thm2

$\vdash \forall\ x$
$\bullet\ SubstForm\ x$
$= (if\ Snd\ x \in Syntax$
$then$
$if\ IsEq\ (Snd\ x) \lor IsMem\ (Snd\ x)$
$then\ SubstAtom\ SubstForm\ x$
$else\ if\ IsTf\ (Snd\ x)$
$then\ SubstTf\ SubstForm\ x$
$else\ \varnothing_g$
$else\ \varnothing_g)$

tv_cases_thm $\vdash \forall\ x \bullet\ x = pTrue \lor x = pFalse \lor x = pU$

tv_distinct_clauses

$\vdash \neg\ pTrue = pFalse$
$\land \neg\ pTrue = pU$
$\land \neg\ pFalse = pTrue$
$\land \neg\ pFalse = pU$
$\land \neg\ pU = pTrue$
$\land \neg\ pU = pFalse$

tran_suborder2_thm

$\vdash trans\ SubOrder2$

tran_suborder2_thm2

$\vdash \forall\ s\ t\ u$
$\bullet\ SubOrder2\ s\ t \land SubOrder2\ t\ u \Rightarrow SubOrder2\ s\ u$

well_founded_SubOrder2_thm

$\vdash well\_founded\ SubOrder2$

evaltf_lemma1

$\vdash \forall\ (v,\ m,\ e)\ ef1\ ef2\ vaf$
$\bullet\ Snd\ vaf \in Syntax$
$\Rightarrow IsTf\ (Snd\ vaf)$
$\Rightarrow (\forall\ y \bullet\ SubOrder2\ y\ vaf \Rightarrow ef1\ y = ef2\ y)$
$\Rightarrow EvalTf\ (v,\ m,\ e)\ ef1\ vaf$
$= EvalTf\ (v,\ m,\ e)\ ef2\ vaf$

evalformfunct_lemma1

$\vdash \forall\ (V,\ \$\in_v,\ \$=_v)$
$\bullet\ EvalFormFunct\ (V,\ \$\in_v,\ \$=_v)\ respects\ SubOrder2$

evalformfunct_fixp_lemma

$\vdash \forall\ (V,\ \$\in_v,\ \$=_v)$
$\bullet\ EvalForm\ (V,\ \$\in_v,\ \$=_v)$
$= EvalFormFunct$
$(V,\ \$\in_v,\ \$=_v)$
$(EvalForm\ (V,\ \$\in_v,\ \$=_v))$

evalformfunct_thm

$\vdash \forall\ (V,\ \$\in_v,\ \$=_v)$
$\bullet\ EvalForm\ (V,\ \$\in_v,\ \$=_v)$

$= (\lambda\ (va,\ f)$
$\bullet$ *if* $f \in$ *WfForms*
then
if *IsEq* $f \vee$ *IsMem* f
then *EvalAtom* ($\$\in_v$, $\$=_v$) $(va,\ f)$
else
EvalTf
$(V,\ \$\in_v,\ \$=_v)$
(*EvalForm* $(V,\ \$\in_v,\ \$=_v)$)
$(va,\ f)$
else $pU)$

evalformfunct_thm2
$\vdash \forall\ (V,\ \$\in_v,\ \$=_v)\ x$
$\bullet$ *EvalForm* $(V,\ \$\in_v,\ \$=_v)\ x$
$=$ (*if* *Snd* $x \in$ *WfForms*
then
if *IsEq* (*Snd* x) $\vee$ *IsMem* (*Snd* x)
then *EvalAtom* ($\$\in_v$, $\$=_v$) x
else
EvalTf
$(V,\ \$\in_v,\ \$=_v)$
(*EvalForm* $(V,\ \$\in_v,\ \$=_v)$)
x
else $pU)$

$\leq_t$_refl_thm $\vdash \forall\ x\bullet\ x \leq_t x$

$\leq_t$_trans_thm
$\vdash \forall\ x\ y\ z\bullet\ x \leq_t y \wedge y \leq_t z \Rightarrow x \leq_t z$

$\leq_t$_clauses $\vdash (\forall\ x\bullet\ pU \leq_t x)$
$\wedge \neg\ pTrue \leq_t pU$
$\wedge \neg\ pFalse \leq_t pU$
$\wedge \neg\ pFalse \leq_t pTrue$
$\wedge \neg\ pTrue \leq_t pFalse$

evalatom_monotone_lemma
$\vdash \forall\ v\ w\ x\ y$
$\bullet\ (v,\ w) \leq_{pr} (x,\ y)$
$\Rightarrow (\forall\ z\bullet$ *EvalAtom* $(v,\ w)\ z \leq_t$ *EvalAtom* $(x,\ y)\ z)$

eqrel_mono_thm
$\vdash \forall\ V\ v\ w\ x\ y$
$\bullet\ (x,\ y) \leq_{pr} (v,\ w)$
$\Rightarrow$ *EqRel* $(V,\ x,\ y) \leq_r$ *EqRel* $(V,\ v,\ w)$

monpprf_memrel_lemma1
$\vdash \forall\ V$
$\bullet$ *MonoPprF* $(\lambda\ (m,\ e)\bullet$ *EvalForm* $(V,\ m,\ e))$
$\Rightarrow$ *MonoPprF*
$(\lambda\ (m,\ e)\ (l,\ r)\bullet$ *MemRel* $(V,\ m,\ e)\ l\ r)$

$\leq_{ts}$_refl_thm
$\vdash \forall\ rs\bullet\ rs \leq_{ts} rs$

$\leq_{ts}$_fc_clauses
$\vdash \forall\ s\ t$
$\bullet\ s \leq_{ts} t$
$\Rightarrow (pTrue \in s \Rightarrow pTrue \in t)$

$\wedge\ (pFalse \in s \Rightarrow pFalse \in t)$
$\wedge\ (pU \in t \Rightarrow pU \in s)$
$\wedge\ (s = \{\} \Leftrightarrow t = \{\})$

mono_evaltf_$\leq_{ts}$_lemma1
$\vdash\ \forall\ s\ t \bullet\ s \leq_{ts} t \Rightarrow s \subseteq \{pTrue\} \Rightarrow t \subseteq \{pTrue\}$

mono_evaltftf_lemma
$\vdash\ \forall\ res1\ res2$
$\bullet\ res1 \leq_{ts} res2 \Rightarrow EvalTf\_tf\ res1 \leq_t EvalTf\_tf\ res2$

evalform_mono_thm
$\vdash\ \forall\ V \bullet\ MonoPprF\ (\lambda\ (m,\ e) \bullet\ EvalForm\ (V,\ m,\ e))$

mono_semanticfunct_lemma1
$\vdash\ \forall\ V$
$\bullet\ MonoPprF\ (\lambda\ (m,\ e) \bullet\ EvalForm\ (V,\ m,\ e))$
$\Rightarrow\ MonoFunct\ (SemanticFunctor\ V)$

mono_semanticfunctor_thm
$\vdash\ \forall\ V \bullet\ MonoFunct\ (SemanticFunctor\ V)$

glb$_{ts}$_thm
$\vdash\ \forall\ tvs$
$\bullet\ (\exists\ tv \bullet\ tv \in tvs)$
$\Rightarrow\ IsLb_{ts}\ tvs\ (glb_{ts}\ tvs)$
$\wedge\ (\forall\ tv \bullet\ IsLb_{ts}\ tvs\ tv \Rightarrow tv \leq_t glb_{ts}\ tvs)$

9 INDEX