Membership Structures

Roger Bishop Jones

Abstract

A queer way of doing set theory in HOL (together with some queer reasons for doing it that way).

Created 2004/07/15

Last Change Date: 2011/10/25 09:10:46

http://www.rbjones.com/rbjpub/pp/doc/t004.pdf

Id: t004.doc,v 1.19 2011/10/25 09:10:46 rbj Exp

Contents

To Do

-

References

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

[2] Roger Bishop Jones. Category Theoretic Foundation Systems. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t018.pdf`.

[3] Roger Bishop Jones. Infinitarily Definable Sets. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t026.pdf`.

[4] ds/fmu/ied/spc002. *HOL Formalised: Semantics*. R.D. Arthan, Lemma 1 Ltd., `http://www.lemma-one.com`.

1 INTRODUCTION

1.1 Original Purposes

This document was originally intended to serve various purposes which I will attempt to describe here under the headings:

- The Foundations of Abstract Semantics.
- Model Construction for Foundation Systems.
- Formalistic Fallacies in Set Theory

1.1.1 Foundations of Abstract Semantics

This is concerned with what can be done formally to progress my understanding of the issues in this area. It remains unclear whether any of the lines which I am considering has any potential.

A foundation for formal semantics is as I conceive it a language and a logic suitable for giving an abstract semantics to a broad range of other languages, and of enabling formal demonstration of truths which are relative to that semantics *analytic* or logically true. The most obvious such language is that of set theory, so this topic is closely connected with the foundations of mathematics.

Here is an indication of some of the questions which seem to me to be of interest in this area and which it may be possibly to progress by formal methods:

- The relationship between languages L1 and L2 which obtains when the semantics of L1 is expressible in L2 is of particular interest. Different such relationships will obtain according to what is held to constitute a semantics (e.g. a truth conditional semantics or a denotational semantics) and according to what counts as expressing a semantics in a language. For the interesting particularisations of this class of partial orders it is of interest whether there is a universal language (in some class of languages under the relevant order) or class of languages (set theory is the most obvious candidate, and it is not immediately obvious whether it could be said to be universal either as a single or as a family of languages).

- The question what are the key features of a language which makes it score high in expressiveness is of interest. It is evident that the ontology of set theory, in particular the kind of entity involved as opposed to the particulars of which entities of that kind exist, is not crucial to expressiveness (other kinds of thing are equally suitable, e.g. functions). On the other hand, it seems possible that the cardinality of the domain of discourse is important at least for sufficiently rich kinds of semantics. But size surely cannot be all?

It is in fact rather more concerned with philosophy than with the brute technical development, and I don't really expect the formal development to get far enough to be valuable in itself.

The philosophy is concerned with set theory as a foundation for abstract semantics, and with the question how set theory can be made sufficiently definite in its meaning to serve well in that role.

Before starting on membership structures I have included an exploration of well-orderings, which may possibly be moved to a separate document at a later date.

1.1.2 Set Theory as a Universal Semantic Foundation System

Set theory is considered here primarily as a candidate universal foundation for semantics, i.e. as a language in which the abstract semantics of any other language may be given and to which, by such means, logical truth in its broadest sense, may be reduced.

1.1.3 An Aside on Logical Truth

Logical truth in its broadest sense I take to be *analyticity* which I also closely associate with necessary truth. Very briefly my rationale in this is as follows. In the first identification I am simply disinclined to follow the novelty of the twentieth century in using the notion of logical truth in as narrow and arbitrary a manner as has become common. You may regard this if you like as an terminological eccentricity (which is how I regard modern usage in this matter). On the relationship between of analyticity with necessity I have more interesting reasons for rejecting the possibility of necessity *de re* (which I take to be any necessity which is not *de dicto*, necessity de dicto being analyticity). If we were free to take any account of the semantics of a language, including some otherwise correct but substantively incomplete account, as the basis for defining the notion of analyticity then we could of course easily contrive examples of necessity *de re*. Take for example the content free semantics for a language, which tells us nothing about which sentences are true or false, and relative to the notion of analyticity (or necessity *de dicto*) which flows from that account all necessity is *de re*. This is of course absurd. To make a judgement about kinds of necessity one must of course judge necessity *de dicto* against a semantics which is complete, at least in its account of the truth conditions for a language. A necessary proposition must however be true *under all conditions*. Any complete account of the truth conditions of the relevant language must of course tell us whenever a statement is unconditionally true. If any statement is judged necessary *de re* it must be that the denial of its being necessary *de re* is based upon an incomplete account of the truth conditions of the language.

There is however one distinction which I would draw between analyticity and necessity. If we think of analyticity as truth in virtue of meaning, and think of propositions as the meanings of indicative sentences, then it is only sentences which can be analytic, which they are if they express a necessary proposition.

All this is far too much on this topic for this context.

1.1.4 Constructing Models for Foundation Systems

This is a fairly concrete aspect which is connected with the more philosophical aspects relating to semantic foundations but not quite so speculative.

I will illustrate this with a brief statement of the actuall application which is driving this part of the work.

I am attempting in [2] the construction of a model for a foundational system whose ontology is category theoretic. This model is obtained in several stages, ultimately from a model of well-founded set theory. The machinery for working with models of set theory is provided by this document.

I have previously done this kind of thing by axiomatising a strong set theory in HOL and then constructing the required models in that set theory with a view to deriving and then applying the theory from there. This is OK for work in these alternative foundational ontologies but is not particularly suitable for meta-theory, and it does not facilitate evaluation of the effects of different set theoretic staring points on the constructed systems.

The idea of working from membership structures instead of from an axiomatisation of set theory is to make the constructions less dependent on any particular set theory, and to enable exploration of therelationship between properties of the initial model of set theory and the various other models constructed from it.

1.1.5 Set Theory Generalised

The kind of "generalisation" which I have in mind here is *without reference to any specific axiomatisation*. The idea is to consider not so much the consequences and the models of particular axiom systems, but rather to consider particular models and the question what properties they posess, and also the properties themselves (including properties which are not expressible in first order logic).

Thus, the interest is not so much in sets as they appear in some theory, but on membership structures and their properties, and how to construct membership structures meeting various desiderata.

1.1.6 Formalistic Fallacies in Set Theory

Formalistic fallacies (of the kind which concern us here) are a species of sceptical fallacy, so I will begin with a few words about sceptical fallacies. Scepticism in radical but consistent forms is a philosophy of doubt, an anti-dogmatic philosophy. In practice is it very common for those of sceptical inclinations not to slip into negative dogmatism, of which the most obvious and extreme form is to assert that nothing can be known.

Formalism was in Hilbert a symptom of naive overenthusiasm about the expressiveness of formal axiomatic systems. He believed that all meaning (at least for mathematics) could be captured axiomatically, and therefore should be. In its manifestation in the culture of mathematical logic today it is a kind of scepticism about semantics. Limitations on the semantic expressiveness of formal languages are now well known, but the formalistic tendency (present in mathematical logic as a whole, not just among those who consider themselves formalists) is to avoid as far as possible the use of concepts which can be formalised or the pursuit of problems which are not formally tractible.

If this were confined to retaining an open mind on those problems which fall beyond the pale that would be OK. These problems beyond the pale might nevertheless be addressed as philosophical problems.

However, the language of mathematical logic slips under this formalistic influence into negative dogmatism. Concepts which should properly involve semantic (definability) are give purely syntactic definitions, denuding the vocabulary with which one might discuss properly semantic questions. Semantics in mathematical logic is reduced to the relationship between syntax and its possible meanings, no longer concerned with identifying the particular meanings which elude formal characterisation. Higher Order Logic is said not to be a logic, but rather a bastard semantic for which no proof system is available, thus perverting at once the notion of logic so that no logic can be incomplete (notwithstanding famous results to the contrary) and semantics, which can not longer express intended or desirable meaning if that could not be matched with a complete proof system.

It seems to me that some of the definite technical results in set theory are spoken of in language which misrepresents their significance, and it is these misrepresentations which I am calling here "formalsitic fallacies".

Here are some prime examples.

Results about what cannot be proven in ZFC, for example about the cardinality of the continuum or other aspects of cardinal exponentialtion, and presented as establishing negative results about what certain cardinals might be.

The relationship between V=L and the existence of measurable cardinals seems to me to be presented in a paradoxical manner.

1.1.7 More Objectives

I have so far poorly stated my objectives, so I will try harder here.

The idea of a semantic approach to set theory as universal semantic foundation system suggests to me that a relationship of interpretability is important. This not the usual rather syntactic notion of interpretability as a relation between theories. The differences I perceive are as follows:

- firstly, the required notion is not interpretability of theoremhood, which gives us an ordering in terms of proof theoretic strength, but of truth. The interpretation must be truth preserving, not derivability preserving.
- secondly, there is an interest in interpretations which not only respect truth, but which respect *meaning*. One may for example complain of an interpretation of set theory in arithmetic, that even if it preserves truth it cannot preserve meaning, since it is of the essence of the meaning of set theory that it is concerned with collections of very large size, and that any interpretation in which all collections are countable (even if there is a paucity of demonstrable bijections to provide evidence of this) cannot be faithful to the meaning of set theory.
- thirdly, all this talk of set *theory* is too syntactic, and wherever possible I am looking to eliminate syntactic in favour of semantics concepts. How to do this remains to be seen, it may be appropriate to consider a language as some collection of properties of membership (or other) structures, and an interpretation as a mapping from properties over one kind of structure to properties of some other kind of structure.

This last point suggests, as is indeed the case, that I am not exclusively concerned with membership structures, though this particular document may remain dedicated to them. I am interested in methods for transforming one foundational ontology into another, for example.

1.2 Other Purposes

The purposes I sought to describe in the previous section have never been significantly progressed, and a lot of the material in this document deserves to be discarded.

However, in the course of my deliberations in relation to non-well-founded set theories, some of the material has been used and additional materials have been included specific to or motivated by work on non-well-founded set theories.

These are:

- the definition of stratified property (section 4.1) and the axiom of stratified comprehension (section 4.2), done while investigating NF/NFU in HOL
- the material on boolean membership structures (section 5), intended for use in finding non-well-founded interpretations but never used
- the material on membership functors (section 6)

It is probable that future development of this document will continue to be along lines suggested by my investigations into non-well-founded set theory, the odds on my returning to the original motivations are not very high.

1.3 Methods

The methods supporting the above objective are still uncertain and evolving. This is not because they are new and original, but because I do not yet understand well enough the pragmatics of doing set theory in ProofPower-HOL along the intended lines.

Partly because the methods are unstable, and partly because of my very limited knowledge of the literature, I am not able to say much about how this relates to what has been done before. On one aspect, the formal definition of semantics using models in set theory without actually adopting an axiomatisation of set theory in HOL, the formal semantics for ProofPower-HOL by Rob Arthan [4] should be mentioned.

The following methodological features can be noted at this point.

- No axioms, no preferred set theory.
- Semantics preferred to syntax, focus on models not theories.

1.3.1 No Axioms

It is this feature which distinguishes the set theory here from my previous formal work in set theory, and which is the reason for starting this document.

Previously I have axiomatised in HOL a higher order version of a first order set theory, typically something like ZFC +inaccessibles and worked with that theory. Either developing that theory (though I have never got very far with that) or using the ontology of that theory in the construction of models for other kinds of foundation system.

The intention here is to avoid commitment to any particular axiomatisation of set theory by talking generally about membership structures and the relationships between their properties. In this approach what might previously have been done by proving a result in an axiomatic set theory would

be done by demonstrating that some property of membership structures (corresponding roughly to the parts of an axiomatisation necessary for the particular result) entails some other property (corresponding to the theorem).

In the case of model constructions, instead of working with an axiomatic system and constructing the model using the ontology of that system (the members of the HOL type which is the domain of the axiomatisation), we take a more external standpoint defining a construction which takes a (more or less arbitrary) membership structure and yields some other kind of structure and showing that the desired properties of target ontology are delivered by the construction provided that certain necessary properties obtain in the membership structure on which the construction is based.

1.3.2 Semantics

2 MEMBERSHIP STRUCTURES

Create new theory "membership".

SML
```
set_flag("pp_use_alias", true);
open_theory "rbjmisc";
force_new_theory "membership";
new_parent "ordered_sets";
new_parent "bin_rel";
set_merge_pcs["hol1", "'ℤ", "'savedthm_cs_∃_proof"];
open RbjTactics1;
```

The following types are used:

MS membership structure

PMS property of membership structure

A membership structure is a set together with a binary relation over that set. This constitutes an interpretation of the language of first order set theory, though not necessarily a model for any particular axiomatisation of set theory such as ZFC.

We are interested in properties of membership structures. We are not exclusively interested in any particular kinds of properties, but we will introduce various classifications of these properties, the first of which will be those which are captured by sentences of first order set theory.

SML
```
declare_type_abbrev ("MS", ["'a"], ⌜:'a SET × ('a → 'a → BOOL)⌝);
declare_type_abbrev ("PMS", ["'a"], ⌜:'a MS → BOOL⌝);
```

SML
```
declare_infix (305, "∈r");
declare_infix (305, "∈a");
declare_infix (305, "∈b");
```

The interest in first order properties is rather subsidiary, the real point of this approach to set theory is to escape from being constrained to first order properties. There is some interest in showing why

this is a good thing, and the notion of first order property is defined primarily in order to be able to show that interesting properties of membership structures often are not first order.

I don't know whether that will prove feasible, its not so hard to prove that properties are first order, but it will probably be a lot harder to prove that properties are not (even where that is pretty obviously the case on the basis of established results like the incompleteness theorems). The most obvious example of a property which is not first order is well-foundedness. The obvious way to prove that well-foundedness is not first order is to prove that if it were we would could get from first order set theory a complete arithmetic. This route however presupposes a proof of the incompleteness of arithmetic, which is not so easy to do formally.

2.1 Operators Over Sets

Most of the operations over sets can be obtained in a canonical manner from a membership structure (even though the structure may not be closed under the operation). This is because typically they are defined extensionally and the membership structures of interest usually are extensional.

There are two exceptions to this. The first is the empty set, which represents the most common point of deviation from extensionality. In a set theory with urelements the urelements will often have no members and are therefore extensionally equivalent to the empty set. This means that you cannot tell from the membership structure alone the identity of the empty set. The second case is the ordered pair constructor. The problem here is that the defining characteristic is not extensional, and there are many ways in which it can be realised extensionally.

The operators over sets will be defined using an abstraction operator which, given some extension returns a set having that extension, if there is one. We also define the function which gives the extension of a member of a membership structure.

HOL Constant

$$\boldsymbol{Ex}: 'a\ MS \to 'a \to 'a\ SET$$

$$\forall\ (X, \$\in_r)\ s\bullet\ Ex\ (X, \$\in_r)\ s = \{y \mid s \in X \wedge y \in X \wedge y \in_r s\}$$

HOL Constant

$$\boldsymbol{Co}: 'a\ MS \to 'a\ SET \to 'a$$

$$\forall\ (X, \$\in_r)\ s\bullet\ (\exists x\bullet\ x \in X\ \wedge\ Ex\ (X, \$\in_r)\ x = s)$$
$$\Rightarrow\ Ex\ (X, \$\in_r)\ (Co\ (X, \$\in_r)\ s) = s$$

We need to be able to talk about membership structures using the wider vocabulary associated with set theory (not just the membership relationship). This of course does not make sense for arbitrary membership relations, which may not be closed under these operators. We could add to and constrain the notion of membership structure to meet this requirement, but instead I propose a function which yields a set of operators which will have the required properties provided that the membership structure has appropriate closure and otherwise will not.

It might convenient if our function delivers a tuple of operators so we define it as follows:

SML

```
declare_infix (310, "∪m");
declare_infix (310, "↾m");
declare_infix (310, "(|)");
```

HOL Constant

$$
\begin{array}{l}
\boldsymbol{MS\_ops}: {'a}\ MS \to ({'a} \\
\qquad \times ({'a} \times {'a} \to {'a}) \\
\qquad \times ({'a} \to {'a} \to {'a}) \\
\qquad \times ({'a} \to {'a}) \\
\qquad \times ({'a} \to {'a}) \\
\qquad \times ({'a} \to ({'a}\ SET) \to {'a}) \\
\qquad \times ({'a} \to ({'a} \to {'a}) \to {'a}) \\
\quad)
\end{array}
$$

$$
\begin{array}{l}
\forall (X{:}{'a}\ SET)\ (\$\in_a{:}{'a} \to {'a} \to BOOL)\ \varnothing_m\ Pair_m\ \$\cup_m\ \bigcup_m\ \mathbb{P}_m\ \$\restriction_m\ \$(\!|\!)\bullet \\
\quad MS\_ops\ (X,\ \$\in_a) = (\varnothing_m,\ Pair_m,\ \$\cup_m,\ \bigcup_m,\ \mathbb{P}_m,\ \$\restriction_m,\ \$(\!|\!)) \\
\quad \Rightarrow (\forall ys\ y\ x\bullet \\
\quad \varnothing_m = Co\ (X,\ \$\in_a)\ \{\} \\
\quad \wedge\ Pair_m\ (x,\ y) = Co\ (X,\ \$\in_a)\ \{x;\ y\} \\
\quad \wedge\ x \cup_m y = Co\ (X,\ \$\in_a)\ ((Ex\ (X,\ \$\in_a)\ x) \cup (Ex\ (X,\ \$\in_a)\ y)) \\
\quad \wedge \bigcup_m x = Co\ (X,\ \$\in_a) \\
\qquad (\bigcup \{z \mid \exists y\bullet\ y \in X \wedge y \in_a x \wedge z = Ex\ (X,\ \$\in_a)\ y\}) \\
\quad \wedge\ \mathbb{P}_m\ x = Co\ (X,\ \$\in_a)\ \{y \mid Ex\ (X,\ \$\in_a)\ y \in (\mathbb{P}\ (Ex\ (X,\ \$\in_a)\ x))\} \\
\quad \wedge\ x \restriction_m ys = Co\ (X,\ \$\in_a)\ \{u \mid u \in X \wedge u \in_a x \wedge u \in ys\})
\end{array}
$$

HOL Constant

$$
\begin{array}{l}
\boldsymbol{union}: {'a}\ MS \to {'a} \to {'a}
\end{array}
$$

$$
\begin{array}{l}
\forall\ (X,\ \$\in_a)\ x\bullet\ union\ (X,\ \$\in_a)\ x = Co\ (X,\ \$\in_a) \\
\qquad (\bigcup \{z \mid \exists y\bullet\ y \in X \wedge y \in_a x \wedge z = Ex\ (X,\ \$\in_a)\ y\})
\end{array}
$$

2.2 Extended Membership Structures

For some applications it seems possible that having just a little more structure will be helpful. The two elements which seem most desirable are the empty set and an ordered pair constructor (with projections).

If there is an empty set, it can of course be identified from the membership relation, but there may be more than one, and it is sometimes important to chose one of them. There are of course ways of defining the ordered pair constructor, but most of the applications I have in mind would rather not have to make the choice.

The following notion of extended membership structure is therefore introduced, and may be subject to adjustments or abandonment in the light of how useful it turns out to be.

There are enough pieces to make use of a HOL labelled product type worthwhile.

HOL Labelled Product

XMS

$Car_x \quad : 'a\ SET;$
$In_x : 'a \to 'a \to BOOL;$
$\varnothing_x : 'a;$
$Op_x \quad : ('a \times 'a) \to 'a;$
$Fst_x \quad : 'a \to 'a;$
$Snd_x \quad : 'a \to 'a;$
$Abs_x \quad : 'a\ SET \to 'a$

The minimal properties one would expect are:

- that the empty set is in the carrier and is empty
- that the ordered pairs are in the carrier and the projections deliver the right values

HOL Constant

$\boldsymbol{is\_XMS}: 'a\ XMS \to BOOL$

$\forall xms{:}'a\ XMS \bullet\ is\_XMS\ xms$
$\Leftrightarrow$
$(\varnothing_x\ xms \in Car_x\ xms \wedge \neg\ \exists s \bullet\ s \in Car_x\ xms \wedge In_x\ xms\ s\ (\varnothing_x\ xms))$
$\wedge \quad (\forall l\ r \bullet\ l \in Car_x\ xms \wedge r \in Car_x\ xms$
$\Rightarrow\ Op_x\ xms\ (l,r) \in Car_x\ xms$
$\wedge\ Fst_x\ xms\ (Op_x\ xms\ (l,r)) = l$
$\wedge\ Snd_x\ xms\ (Op_x\ xms\ (l,r)) = r)$
$\wedge \quad (\forall s{:}'a\ SET \bullet\ (\exists t{:}'a \bullet\ \forall u{:}'a \bullet\ u \in s \Leftrightarrow (In_x\ xms)\ u\ t) \Rightarrow$
$(\forall u{:}'a \bullet\ u \in s \Leftrightarrow (In_x\ xms)\ u\ (Abs_x\ xms\ s)))$
$\wedge \quad WellFounded\ (Car_x\ xms,\ In_x\ xms)$

3 BASIC PROPERTIES OF MEMBERSHIP RELATIONS

The term "membership relation" is use purely to indicate motivation, this is not a non trivial subclass of binary relationships.

The following are key properties which yield our broadest classification of membersip relations:

1. extensionality
2. purity
3. well-foundedness

3.1 Extensionality

None of these properties is essential for a membership relation. The one which is perhaps most conspiciously characteristic of set membership is extensionality, but constructive set theories are likely not to be extensional, and set theories with urelements may have a qualified extensionality (as in NFU).

A set theory is "pure" if everything in the domain of discourse is a set. This is connected with extensionality, since non-sets have no members and are therefore extensionally the same as the empty set (except for Quine atoms which are their own unit set, and permit something like urelements in a fully extensional theory).

We are concerned here only with relations whose type is some instance of $\ulcorner : 'a \to 'a \to BOOL \urcorner$ and so there is only one type of entity in the domain of discourse, and so in some sense everything is a set. However, if extensionality is weakened there may be any number of sets which have no members, and we may think of the extra ones as urelements. If well-foundedness is absent or qualified then Quine's trick of treating urlements as their own unit set allows full extensionality to be retained in a system in which some things are not thought of as sets.

HOL Constant

$$\boldsymbol{extensional}: 'a\ PMS$$

$$\forall\ (X,\ \$\in_r)\bullet\ extensional\ (X,\ \$\in_r) \Leftrightarrow \forall s\ t\bullet\ s \in X \wedge t \in X \Rightarrow (s = t \Leftrightarrow (\forall\ u\bullet\ u \in X \Rightarrow (u \in_r s \Leftrightarrow u \in_r t)))$$

The following weakend extensionality is satisfied by models of NFU:

HOL Constant

$$\boldsymbol{weakly\_extensional}: 'a\ PMS$$

$$\forall\ (X,\ \$\in_r)\bullet\ weakly\_extensional\ (X,\ \$\in_r) \Leftrightarrow \forall s\ t\bullet\ s \in X \wedge t \in X \wedge (\exists v\bullet\ v \in X \wedge v \in_r s) \Rightarrow (s = t \Leftrightarrow (\forall\ u\bullet\ u \in X \Rightarrow (u \in_r s \Leftrightarrow u \in_r t)))$$

Any membership relation can be collapsed into one which is extensional, though in the worst case it might be trivial, by forming equivalence classes and redefining the relationship over those classes.

We obtain the new membership structure by taking the domain to be equivalence classes of members under some equivalence relation and then defining a new membership operation over those classes as follows:

It is convenient to use the theory *equiv_rel* in which equivalences are curried binary relations rather than sets of ordered pairs (as in theory *bin_rel*). We need to take an intersection of a collection of such relations so this is defined first.

HOL Constant

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

$$\forall\ sr\bullet\ \wedge_r\ sr = \lambda x\ y\bullet\ \forall z\bullet\ z \in sr \Rightarrow z\ x\ y$$

Then *ExtQuot* gives the set of sets of elements which must be identified to give an extensional relationship.

HOL Constant

$$\textbf{\textit{ExtQuot}}: 'a\ MS \to 'a\ SET\ SET$$

$$\forall\ (X,\ \$\in_r)\bullet\ ExtQuot\ (X,\ \$\in_r) =$$
$$QuotientSet\ X\ (\wedge_r\ \{r : 'a \to 'a \to BOOL$$
$$\mid\ Equiv\ (X,\ r)$$
$$\wedge\ \forall x\ y\bullet\ x \in X \wedge y \in X \wedge ((\forall z\bullet\ z \in_r x \Leftrightarrow z \in_r y)$$
$$\Rightarrow\ r\ x\ y)\})$$

Finally we define the relationship:

HOL Constant

$$\textbf{\textit{ExtRel}}: 'a\ MS \to 'a\ SET\ MS$$

$$\forall\ (X,\ \$\in_r)\bullet\ ExtRel\ (X,\ \$\in_r) =$$
$$let\ Y = ExtQuot\ (X,\ \$\in_r)$$
$$in\ (Y,\ \lambda l\ r\bullet\ \exists u\ v\bullet\ u \in l \wedge v \in r \wedge u \in_r v)$$

3.2 Well-Foundedness

The definition of well-foundedness used here is the concept *WellFounded* from the theory "ordered_sets".

Set theories whose intuition is based in the cumulative or iterative heirarchy are both extensional and well-founded, and I will be concerned for the time being with just these kinds of membersbip relation. These two properties can be thought of as telling us what kind of thing a set is, and the rest of the axioms in a systems like ZFC, possibly with large cardinal axioms, are trying to maximise the how many of the things with these properties can be shown to be in the domain of discourse.

This explanation connects with the usual description of the cumulative heirarchy, in which the formation of the domain of discourse of set theory is presented as occurring in stages. At each stage one adds to the cumulative hierarchy all the sets which can be formed from those already obtained (the first "all") and then we are to understand that one goes through as many stages as one possibly can (the second "all"), thereby aggregating all the pure well-founded collections. Unfortunately, if one ever could complete this process the result would be a well-founded collection which would not contain itself, and so we must suppose that the construction cannot be completed.

My present interest is therefore, at present, in considering how to formalise the description of larger and larger parts of this incompletable construction.

The decomposition of the "all" in the intuitive concept suggests that the formalisation may be able to treat these two "all"s separately. The idea is that we can say that at each stage all possible new subsets are created, and then consider how to say that the number of stages is very large, separating questions of width and height. A clean separation turns out not to be possible in a first order axiomatisation, but in the present context, i.e. in a higher order logic construed under the standard semantics, it is possible (though perhaps more by theft than honest toil, the relevant notion of "all" being readily expressed under the standard semantics).

However, there are some preliminaries which seem to be necessary to adeqately address either "all". To get "full-width" we want to say that at each stage all the subsets of the previous stage become sets. For this we need to be able to talk about the stages.

3.3 Order Morphisms etc.

HOL Constant

$$\textbf{\textit{homomorph}}: ('a\ MS \times ('b\ MS)) \to ('a \to 'b) \to BOOL$$

$$\forall\ (X, \$\in_a):'a\ MS;\ (Y, \$\in_b):'b\ MS;\ f:'a \to 'b\bullet$$
$$homomorph\ ((X, \$\in_a), (Y, \$\in_b))\ f \Leftrightarrow\ \forall s\ t:'a\bullet\ s \in X \wedge t \in X \wedge s \in_a t$$
$$\Rightarrow (f\ s) \in Y \wedge (f\ t) \in Y \wedge (f\ s) \in_b (f\ t)$$

An embedding is not just an injective homomorphism:

HOL Constant

$$\textbf{\textit{embedding}}: ('a\ MS \times ('b\ MS)) \to ('a \to 'b) \to BOOL$$

$$\forall\ (X, \$\in_a):'a\ MS;\ (Y, \$\in_b):'b\ MS;\ f:'a \to 'b\bullet$$
$$embedding\ ((X, \$\in_a), (Y, \$\in_b))\ f \Leftrightarrow\ OneOne\ f$$
$$\wedge\ \forall s\ t:'a\bullet\ s \in X \wedge t \in X \Rightarrow (f\ s) \in Y \wedge (f\ t) \in Y \wedge (s \in_a t \Leftrightarrow (f\ s) \in_b (f\ t))$$

A substructure is something which embeds in the above sense.

SML

```
declare_infix (300, "substructure_of");
```

HOL Constant

$$\$\textbf{\textit{substructure\_of}}: ('a\ MS) \to ('a\ MS) \to BOOL$$

$$\forall\ (X, \$\in_a)\ (Y, \$\in_b)\bullet$$
$$(X, \$\in_a)\ substructure\_of\ (Y, \$\in_b) \Leftrightarrow X \subseteq Y \wedge \forall s\ t:'a\bullet\ s \in X \wedge t \in X \Rightarrow (s \in_a t \Leftrightarrow s \in_b t)$$

Since the sets in a substructure need not have the same extension as those in the extension, we have a stronger notion here which corresponds to inclusion between transitive models in preserving the extension of the original sets.

SML

```
declare_infix (300, "xsubstr");
```

HOL Constant

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

$$\forall\ (X, \$\in_a)\ (Y, \$\in_b)\bullet$$
$$(X, \$\in_a)\ xsubstr\ (Y, \$\in_b) \Leftrightarrow X \subseteq Y \wedge \forall s\ t\bullet\ t \in X \wedge s \in Y \Rightarrow (s \in_a t \Leftrightarrow s \in_b t)$$

SML

```
declare_infix (310, "◁m");
```

HOL Constant

$$\$\textbf{\textit{◁m}}: ('a \to BOOL) \to 'a\ MS \to 'a\ MS$$

$$\forall\ P\ (X, \$\in_a) \bullet P \lhd_m (X, \$\in_a) = (X \cap \{x \mid P\ x\}, \$\in_a)$$

Membership structures need not be extensional. and so there may not be a unique set with no elements. It is convenient to be able to talk about the empty things.

HOL Constant

$\$\boldsymbol{ms}\_\varnothing$: $'a\ MS \to 'a \to BOOL$

$\forall\ (X,\ \$\in_a)\ s\bullet\ ms\_\varnothing\ (X,\ \$\in_a)\ s$
$\quad \Leftrightarrow\ s \in X \wedge \forall z\bullet\ z \in X \Rightarrow \neg\ z \in_a s$

3.4 Ordinals

The reason for treating ordinals here is as follows. One of the important properties of structures is their height. Strong set theories are those with large cardinal axioms, which appear to be ensisting on the existence of very large sets, but which as is well know, do not succeed in their objective. Set theories with large cardinal axioms still have countable models. This is because *true* large cardinal properties are not expressible in first order logic, they are not first order properties of membership structures (as defined below).

So that we can talk about models which really do have large sets in them we need to define large cardinal *properties* as distinct from large cardinal axioms. It is desirable to do this independently of any particular axiomatisation of set theory.

In our present context we could treat cardinals as equivalence classes of equipollent collections, but I propose nonetheless to follow more closely their treatment in well-founded set theories by taking them to be *alephs*, those ordinals which are larger than all their predecessors.

The stages can be ordered using ordinals, and so first we need a concept of ordinal number. We can think of ordinals in terms of width, for the ordinals are among the membership relations of least width (which are those in which only one new set is introduced at each stage).

There are two kinds of (representation or realisation of) ordinals of interest here. The first kind of ordinal is an element in the domain of a membership relation. The second is a membership relation of which the domain as a whole is an ordinal. In this case all the elements in the domain are ordinals and the relation of a whole may be thought of as the class of these ordinals.

Subscripts will be used to distinguish properties of ordinal relations from the corresponding property of ordinal elements.

A *relation* is "ordinal" iff it is a transitive strict well-ordering. The concept of *WellOrdering* in the theory n*ordered_sets* does not require strictness so we define an ordinal using $WellFounded$ (which does require strictness) and $LinearOrder$.

HOL Constant

$\boldsymbol{Ordinal_{ms}}$: $'a\ PMS$

$\forall\ ms\bullet\ Ordinal_{ms}\ ms \Leftrightarrow LinearOrder\ ms \wedge Trans\ ms \wedge WellFounded\ ms$

An ordinal set is one which is transitive and well ordered by membership.

HOL Constant

$\subseteq_{\boldsymbol{ms}}$: $'a\ MS \to 'a \to 'a \to BOOL$

$\forall\ (X,\ \$\in_r)\!:'a\ MS;\ a\ b\!:'a\bullet \subseteq_{ms}\ (X,\ \$\in_r)\ a\ b \Leftrightarrow \forall x\bullet\ x \in X \wedge x \in_r a \Rightarrow x \in_r b$

SML

$declare\_alias\ (\texttt{"}\subseteq\texttt{"},\ \ulcorner\subseteq_{ms}\urcorner);$

HOL Constant

$$\boldsymbol{TransitiveSet}: 'a\ MS \to 'a \to BOOL$$

$$\forall\ (X,\ \$\in_r):'a\ MS;\ a:'a\bullet\ TransitiveSet\ (X,\ \$\in_r)\ a \Leftrightarrow \forall x\bullet\ x \in X \wedge x \in_r a \Rightarrow \subseteq_{ms}\ (X,\ \$\in_r)\ x\ a$$

The following function delivers the restriction of a membership relation to the extension of one of its sets.

HOL Constant

$$\boldsymbol{set2rel}: 'a\ MS \to 'a \to 'a\ MS$$

$$\forall\ (X,\ \$\in_r):'a\ MS;\ a:'a\bullet\ set2rel\ (X,\ \$\in_r)\ a = (\{x:'a \mid x \in_r a \wedge x \in X\},\ \$\in_r)$$

We then define the ordinals sets as those sets which are transitive and well-ordered by membership.

HOL Constant

$$\boldsymbol{Ordinal}: 'a\ MS \to 'a \to BOOL$$

$$\forall\ ms\ s\bullet\ Ordinal\ ms\ s \Leftrightarrow TransitiveSet\ ms\ s \wedge WellOrdering\ (set2rel\ ms\ s)$$

HOL Constant

$$\boldsymbol{ordinals}: 'a\ MS \to 'a\ SET$$

$$\forall\ ms\bullet\ ordinals\ ms = \{s \mid Ordinal\ ms\ s\}$$

3.5 Ordinal Arithmetic

In this section I am using *Co* freely, in effect assuming extensionality in the membership structure.

First, zero.

HOL Constant

$$\varnothing: 'a\ MS \to 'a$$

$$\forall\ (X,\ \$\in_a)\bullet\ \varnothing\ (X,\ \$\in_a) = Co\ (X,\ \$\in_a)\ \{\}$$

Next the successor relationship:

HOL Constant

$$\boldsymbol{Succ_o}: 'a\ MS \to 'a \to 'a \to BOOL$$

$$\forall\ ms\ a\ b\bullet\ Succ_o\ ms\ a\ b \Leftrightarrow Ex\ ms\ b = \{x \mid x = a \vee x \in Ex\ ms\ a\}$$

HOL Constant

$$\boldsymbol{SuccClosed_o}: 'a\ MS \to BOOL$$

$$\forall\ ms\bullet\ SuccClosed_o\ ms \Leftrightarrow \forall x\bullet\ Ordinal\ ms\ x \Rightarrow \exists y\bullet\ Succ_o\ ms\ x\ y$$

Then the successor function:

HOL Constant

$$\boldsymbol{succ_o}: 'a\ MS \to 'a \to 'a$$

$$\forall\ (X, \$\in_a)\ x\bullet\ succ_o\ (X, \$\in_a)\ x = Co\ (X, \$\in_a)\ \{y \mid y \in X \wedge (y = x \vee y \in_a x)\}$$

HOL Constant

$$\boldsymbol{pred_o}: 'a\ MS \to 'a \to 'a$$

$$\forall\ ms\ x\bullet\ Ordinal\ ms\ x \Rightarrow pred_o\ ms\ (succ_o\ ms\ x) = x$$

Addition is defined using transfinite recursion.

HOL Constant

$$\boldsymbol{\$sum_o}: 'a\ MS \to 'a \to 'a \to 'a$$

$$\forall\ ms\ x\ y\bullet\ \{x;y\} \subseteq ordinals\ ms \Rightarrow sum_o\ ms\ x\ (\varnothing\ ms) = x \wedge\ sum_o\ ms\ x\ (succ_o\ ms\ y) = succ_o\ ms\ (sum_o\ ms\ x\ y)$$

3.6 Ordinals in Extended Membership Structures

The VonNeuman ordinals can be extracted from an XMS as follows:

HOL Constant

$$\boldsymbol{xms\_ordinals}: 'a\ XMS \to 'a\ SET$$

$$\forall xms\bullet xms\_ordinals\ xms = \{x \mid x \in Car_x\ xms \wedge Ordinal\ (Car_x\ xms, In_x\ xms)\ x\}$$

3.7 WK Ordered Pairs

The following relation tells us when something in the domain of a relationship structure is an ordered pair.

HOL Constant

$$\boldsymbol{\$ms\_pair}: 'a\ MS \to ('a \times 'a) \to 'a \to BOOL$$

$$\forall\ (X, \$\in_a)\ (l,r)\ p\bullet\ ms\_pair\ (X, \$\in_a)\ (l,r)\ p \Leftrightarrow p \in X \wedge \forall z\bullet\ z \in X \Rightarrow (z \in_a p \Leftrightarrow z = l \vee z = r)$$

HOL Constant

$\$\boldsymbol{ms\_unit}$: $'a\ MS \to 'a \to 'a \to BOOL$

$\forall\ ms\ s\ u\bullet\ ms\_unit\ ms\ s\ u$
$\quad \Leftrightarrow\ ms\_pair\ ms\ (s,s)\ u$

HOL Constant

$\$\boldsymbol{ms\_wkp}$: $'a\ MS \to ('a \times 'a) \to 'a \to BOOL$

$\forall\ ms\ (l,r)\ p\bullet\ ms\_wkp\ ms\ (l,r)\ p$
$\quad \Leftrightarrow \exists s\ t{:}'a\bullet\ ms\_unit\ ms\ l\ s \wedge ms\_pair\ ms\ (l,\ r)\ t \wedge ms\_pair\ ms\ (s,t)\ p$

3.8 Standard Models

The use of *standard* here follows its use in the context of higher order logic, in which a standard model is one in which the power sets are complete. In set theory this comes down to the idea that the standard models are the $V(\alpha)$ for some ordinal α.

1

We therefore define the relationship V between ordinals and standard models of rank α. The relationship is defined by transfinite recursion, and we therefore need to use a suitable recursion theorem to establish its consistency.

First of all we will have to defined certain elementary concepts. The definition informally is:

$V\ \alpha = \bigcup_{\{\beta\ <\ \alpha\}} \mathbb{P}(V\ \beta)$

The universe of rank α is the union of the power sets of the universes of all smaller ranks.

or:

$V\ \alpha = \{x \mid \exists \beta < \alpha\bullet\ x \subseteq V\ \beta\}$

To formalise this we need to a functor over which this is a fixed point.

HOL Constant

$\boldsymbol{Vf}$: $((('b \to 'b \to BOOL) \times 'a\ MS \to 'a) \to (('b \to 'b \to BOOL) \times 'a\ MS \to 'a)$

$\forall f\bullet\ Vf\ f =\ \lambda(\$<<,\ ms)\bullet$
$\quad let\ (\varnothing_m,\ Pair_m,\ \$\cup_m,\ \bigcup_m,\ \mathbb{P}_m,\ \$\restriction_m,\ \$(\!|\!|)) = MS\_ops\ ms$
$\quad in\ Co\ ms\ \{z\ \mid z \in Fst\ ms$
$\qquad \wedge \exists v\bullet\ (Ex\ ms\ z) \subseteq (Ex\ ms\ (f\ ((\lambda x\ y\bullet\ x << y \wedge y << v),\ ms)))\}$

Using a recursion theorem we derive from this the condition necessary to justify the following definition of V:

[1] There is a weaker notion of standard in set theory which originates I believe in the theory of forcing. This effectively enforces only well-foundedness, a standard model as defined by Cohn is just a transitive set.

$recursion\_theorem =$
$\vdash \forall ((X : 'a\ SET), (\$<< : 'a \to 'a \to BOOL))\ (G : ('a \to 'b) \to 'a \to 'b)\bullet$
$FunctRespects\ G\ (X, \$<<) \wedge WellFounded\ (X, \$<<)$
$\Rightarrow UniquePartFixp\ X\ G : THM$

4 FIRST ORDER PROPERTIES

In order to define the notion of a "first order property" we need to mimic the satisfaction relation. Sentences define sets of interpretations, but formulae only do so in the context of an assignment to the free variables in the formula. The operators we define therefore operate over parameterised properties of interpretations.

It proves desirable to make this theory polymorphic not only in the type of the sets involved, but also in the notion of variable (sometimes we may wish to work with assignments to terms). An assignment to free variables is modelled as a function from objects of type ⌜:'v⌝ (a type variable) to elements in the domain of the relation (here ⌜:'s⌝).

SML

$declare\_type\_abbrev\ (\texttt{"}\boldsymbol{VA}\texttt{"}, [\texttt{"}'s\texttt{"}, \texttt{"}'v\texttt{"}], \ulcorner :'v \to 's \urcorner);$

HOL Labelled Product

PPMS

$ppms : ('s, 'v)\ VA \to 's\ PMS$

Various operators over properties of membership relations are now defined.

The language of set theory which we consider has the following simplifying characteristics:

1. the only terms are variables
2. the relations are equality and membership
3. the propositional connectives are conjunction and negation
4. there is just one, existential, quantifier

Membership statements are parameterised by the names of the two variables between which membership is predicated, and the parameterised property of membership relations which they denote is:

SML

$declare\_infix\ (305, \texttt{"}\in_p\texttt{"});$

HOL Constant

$\$\in_{\boldsymbol{p}}: 'v \to 'v \to ('s, 'v)\ PPMS$

$\forall\ s\ t{:}'v\bullet\ (s \in_p t) = MkPPMS\ (\lambda va{:}\ ('s, 'v)\ VA; (X, \$\in_r){:}'s\ MS\bullet$
$(va\ s) \in_r (va\ t))$

Equality is defined similarly.

SML

$$declare\_infix\ (305,\ "=_p");$$

HOL Constant

$$\$=_{\boldsymbol{p}}: 'v \to 'v \to ('s, 'v)\ PPMS$$

$$\forall\ s\ t{:}'v\bullet\ s =_p\ t = MkPPMS(\lambda\ va{:}('s, 'v)\ VA;\ (X, \$\in_r){:}\ 's\ MS\bullet (va\ s) = (va\ t))$$

Conjunction denotes a binary operation over PPMS's.

SML

$$declare\_infix\ (302,\ "\wedge_p");$$

HOL Constant

$$\$\wedge_{\boldsymbol{p}}: ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$$

$$\forall\ l\ r\bullet\ l \wedge_p\ r = MkPPMS(\lambda va\ (X, \$\in_r)\bullet\ (ppms\ l\ va\ (X, \$\in_r)) \wedge (ppms\ r\ va\ (X, \$\in_r)))$$

Negation denotes a monadic operator.

HOL Constant

$$\neg_{\boldsymbol{p}}: ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$$

$$\forall\ f\bullet\ \neg_p\ f = MkPPMS(\lambda va\ (X, \$\in_r)\bullet\ \neg\ (ppms\ f\ va\ (X, \$\in_r)))$$

The existential quantifier binds a variable and its value depends upon assigning values to that variable in a variable assignment. The following function modifies a variable assignment.

HOL Constant

$$\boldsymbol{assign}:\ 's \to 'v \to ('s, 'v)\ VA \to ('s, 'v)\ VA$$

$$\forall\ a\ s\ va\ t\ \bullet\ assign\ a\ s\ va\ t = if\ t = s\ then\ a\ else\ va\ t$$

Finally the existential quantifier, which is parameterised by the name of the variable it binds and the value of the formula it quantifies over.

HOL Constant

$$\exists_{\boldsymbol{p}}: 'v \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$$

$$\forall\ s\ f\bullet\ \exists_p\ s\ f = MkPPMS(\lambda va\ (X, \$\in_r)\bullet\ \exists a\bullet\ a \in X \wedge ppms\ f\ (assign\ a\ s\ va)\ (X, \$\in_r))$$

Next we define the set of parameterised properties which can be constructed by these operators and then take their closures to eliminate the parameters.

HOL Constant

$$\boldsymbol{fof}: ('s, 'v)\ PPMS \to BOOL$$

$$\forall ppms1 \bullet fof\ ppms1 \Leftrightarrow$$
$$\quad \forall pppms \bullet (\forall s1\ s2\ p1\ p2 \bullet$$
$$\qquad pppms\ (s1 \in_p s2)$$
$$\qquad \wedge\ pppms\ (s1 =_p s2)$$
$$\qquad \wedge\ (pppms\ p1 \wedge pppms\ p2$$
$$\qquad\quad \Rightarrow pppms\ (p1 \wedge_p p2)$$
$$\qquad\quad \wedge\ pppms\ (\neg_p p1)$$
$$\qquad\quad \wedge\ pppms\ (\exists_p s1\ p1))$$
$$\qquad)$$
$$\qquad \Rightarrow pppms\ ppms1$$

Finally we define the property of properties of membership structures which consists in being definable by a set of sentences of first order set theory.

First we define the closure of a first order formula. Note that this function serves to convert a paramaterised property of membership structures into a plain property of membership structures, and this is its primary purpose. The fact that it also effectively undertakes a universal closure is a side effect (though this will simplify some demonstrations), since it is primarily intended for application to closed formulae.

An awkwardness arises here in dealing with the possibility of empty structures. An interpretation of a first order language must be non-empty, but for present purposes it is convenient for the notion of a first order property of structures to take into account the possibility of empty structures.

HOL Constant

$$\boldsymbol{foc}: ('s, 'v)\ PPMS \to 's\ PMS$$

$$\forall ppms1\ (X, \$\in_m) \bullet foc\ ppms1\ (X, \$\in_m) \Leftrightarrow$$
$$\quad \forall va \bullet X = \{\} \vee (\forall v \bullet va\ v \in X)$$
$$\qquad \Rightarrow ppms\ ppms1\ va\ (X, \$\in_m)$$

The concept of first order sentence, as here presented using semantic rather than syntactic entities, is in fact the property of being a property of membership structures which can be expressed by a sentence in the language of first order set theory. The name *fop* for *first order property* is therefore chosen rather than *fos* for *first order sentence*.

This property is independent of the type of variables, but is defined in terms of entities which are polymorphic in the type of variables. For the definition to be conservative we have to parameterise it with a variable, which is ignored. In applications this definition can be instantiated to the application type to yield a property.

HOL Constant

$$\boldsymbol{fop}: 'v \to 's\ PMS \to BOOL$$

$$\forall pmr\ v \bullet fop\ v\ pmr \Leftrightarrow \exists f: ('s, 'v)\ PPMS \bullet fof\ f \wedge pmr = foc\ f$$

For convenience the following derived constructors are defined:

SML

```
declare_infix (298, "∨p");
declare_infix (295, "⇒p");
declare_infix (290, "⇔p");
```

HOL Constant

$\$\vee_{\boldsymbol{p}} : ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$

$\forall l\ r\bullet\ l \vee_p r = \neg_p (\neg_p l \wedge_p \neg_p r)$

HOL Constant

$\$\Rightarrow_{\boldsymbol{p}} : ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$

$\forall l\ r\bullet\ l \Rightarrow_p r = r \vee_p \neg_p l$

HOL Constant

$\$\Leftrightarrow_{\boldsymbol{p}} : ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$

$\forall l\ r\bullet\ l \Leftrightarrow_p r = (l \Rightarrow_p r) \wedge_p (r \Rightarrow_p l)$

HOL Constant

$\forall_{\boldsymbol{p}}: 'v \to ('s, 'v)\ PPMS \to ('s, 'v)\ PPMS$

$\forall\ s\ p \bullet \forall_p\ s\ p = \neg_p (\exists_p\ s\ (\neg_p\ p))$

4.1 Stratified Properties

Some set theories, notably Quine's NF and NFU, are defined using stratified comprehension rather than the more usual *separation*. Stratification is a syntactic constraint on the formulae of set theory which can be used to define a set. This limited class of formulae defines a subset of the first order properties of membership structures, which we will call stratified first order properties.

The *semantics* of formulae is unaffected by this constraint, so in defining a stratified property exactly the same constuctors can be used as for non-stratified formulae. However, when two properties are combined, there must be a check imposed to establish that the combined property is stratified. This only happens in conjunction, so only one check need be specified. The check is that there is a type assignment to the free variables of the formula, and this is best performed using a type assignment to the formulae to be combined. The type assignments are therefore produced intially for the atomic formulae, and then elaborated by each syntactic construction, with a possibility of failure on conjunction.

4.1.1 Type Assignment

This version differs from the previous in the following respects:

- A type assignment is represented as a set of lists of sets of variable names

- The specification for a stratified property uses a single type assignment given ab initio rather than building up a type assignement as the property is built from the primitive relations.
- The type assignment does not take account of bound variables, and this means that some terms in which the same variable is used more than once in quantifiers there may not be a type assignment adequate for these purpose. In these cases it will be necessary to rename some bound variables before a type assignment becomes available. Because the type assignment comes top down it must contain information about the types of bound variables, which was not the case in the previous version, and since we have not made provision for scoping such information a proper description of the types may not be possible where the same name is bound more than once.

SML

$declare\_type\_abbrev(\texttt{"}\boldsymbol{TA}\texttt{"},\ [\texttt{"}'v\texttt{"}],\ \ulcorner:((\prime v\ SET)LIST)LIST\urcorner);$

In the type assignment each list is a list of sets of type-related variable names. The members of each set all have the same type, the members of successive sets have successive types. Separate lists are each independent of each other. For example the formula ⌜x ∈ y ∧ b ∈ y ∧ v = w ∧ w ∈ u⌝ would have the type assignment ⌜ [[{x; b}; {y}]; [{v; w}; {u}]] ⌝.The well-formedness conditions are simply that no name occurs more than once, i.e. that the sets in each list are pairwise disjoint and the unions of the sets in each list are pairwise disjoint. To test an atomic formula for correctness relative to a type assignment, restrict the assignment to the names in the formula, then you should have, for ⌜a ∈ b⌝, ⌜[[{a};{b}]]⌝ and for a = b, ⌜[[{a;b}]]⌝ (bearing in mind that order in list displays is signficant but in set displays is not.

To check a formula you need only check each atomic sentence which occurs in it.

Now we formalise this.

HOL Constant

$$\boldsymbol{Disjoint}: \prime a\ SET\ LIST \to BOOL$$

$$(Disjoint\ [] \Leftrightarrow T)$$
$$\wedge\ \forall\ h\ t\bullet\ Disjoint\ (Cons\ h\ t) \Leftrightarrow ((h \cap (ListUnion\ t) = \{\}) \wedge Disjoint\ t)$$

HOL Constant

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

$$\forall\ f\ s\bullet\ SetMap\ f\ s = \{x \mid \exists y\bullet\ y \in s \wedge x = f\ y\}$$

HOL Constant

$$\boldsymbol{TaDom}: \prime v\ TA \to \prime v\ SET$$

$$\forall\ ta\bullet\ TaDom\ ta = ListUnion\ (Map\ ListUnion\ ta)$$

HOL Constant

$$\boldsymbol{WfTa}: \prime v\ TA \to BOOL$$

$$\forall ta\bullet\ WfTa\ ta \Leftrightarrow$$
$$\forall_L\ (Map\ Disjoint\ ta)$$
$$\wedge\ Disjoint\ (Map\ ListUnion\ ta)$$

$$ListUnion\_thm = \vdash ListUnion\ [] = \{\} \wedge (\forall\ h\ t\bullet\ ListUnion\ (Cons\ h\ t) = h \cup ListUnion\ t)$$

SML

$$declare\_infix\ (310,\ "\lhd_{ta}");$$

HOL Constant

$$\$\lhd_{\boldsymbol{ta}}: 'v\ SET \to 'v\ TA \to 'v\ TA$$

$$\forall\ vs\ ta\bullet\ vs \lhd_{ta} ta = Map\ (\lambda l\bullet\ Map\ (\$\cap\ vs)\ l)\ ta$$

We do not need functions to construct these type assignments, we need only to be able to check properties against them. In fact we only need to check the atomic propositions.

HOL Constant

$$\boldsymbol{Check\_eq}: 'v\ TA \to 'v \to 'v \to BOOL$$

$$\forall\ ta\ a\ b\bullet\ Check\_eq\ ta\ a\ b \Leftrightarrow \{a;\ b\} \lhd_{ta} ta = [[\{a;b\}]]$$

HOL Constant

$$\boldsymbol{Check\_\in}: 'v\ TA \to 'v \to 'v \to BOOL$$

$$\forall\ ta\ a\ b\bullet\ Check\_\in\ ta\ a\ b \Leftrightarrow \{a;\ b\} \lhd_{ta} ta = [[\{a\};\{b\}]]$$

HOL Constant

$$\boldsymbol{wtfof}: 'v\ TA \to ('s, 'v)PPMS \to BOOL$$

$$\begin{array}{l}
\forall ppms1\ ta\bullet\ wtfof\ ta\ ppms1 \Leftrightarrow \\
\quad \forall pppms\bullet\ (\forall s1\ s2\ p1\ p2\bullet \\
\qquad (Check_\in\ ta\ s1\ s2 \Rightarrow pppms\ (s1 \in_p s2)) \\
\qquad \wedge\ (Check_eq\ ta\ s1\ s2 \Rightarrow pppms\ (s1 =_p s2)) \\
\qquad \wedge\ (pppms\ p1 \wedge pppms\ p2 \\
\qquad\quad \Rightarrow pppms\ (\neg_p\ p1) \\
\qquad\quad \wedge\ pppms\ (\exists_p\ s1\ p1) \\
\qquad\quad \wedge\ pppms\ (p1 \wedge_p p2)) \\
\qquad) \\
\qquad \Rightarrow pppms\ ppms1
\end{array}$$

We are now in a position to define the notion of a stratified first order property.

HOL Constant

$$\boldsymbol{StratProp}: ('s, 'v)PPMS \to BOOL$$

$$\forall ppms1\bullet\ StratProp\ ppms1 \Leftrightarrow \exists ta\bullet\ WfTa\ ta \wedge wtfof\ ta\ ppms1$$

My interest in stratified first order properties differs from that in arbitrary first order properties. My interest is in their use in determining the extension of sets and thence in properties of stratified abstraction for non-well founded set theories (such as NF and NFU).

There are two ways in which I propose to apply the concepts. The primary consideration so far as this document is concerned is in formalising the property of membership which consists in closure under stratified abstraction. However, I also want to export to another document in which I am exploring the formalisations of NF and NFU a property suitable for use in expressing an axiom of stratified abstraction.

Lets look first at the first of these and then consider how the second requirement can be satisfied.

To realise the first we need to abstract over a variables (which may or may not actually have free occurrences in the formula corresponding to the property at hand). This of course is a set abstraction i.e. a comprehension. It should convert a parameterised property of membership systems to another property which is true of those membership systems in which there exists a set with an appropriate extension.

HOL Constant

$$\boldsymbol{Comp}: 'v \to ('s,'v)PPMS \to ('s,'v)PPMS$$

$$\begin{array}{l}
\forall v\ ppms1 \bullet\ Comp\ v\ ppms1 = MkPPMS \\
\quad (\lambda va\ (V, \$\in_r) \bullet \\
\qquad \exists s \bullet\ s \in V \wedge \forall t \bullet\ t \in V \Rightarrow \\
\qquad\quad (t \in_r s \Leftrightarrow \\
\qquad\quad ppms\ ppms1\ (\lambda w \bullet\ if\ w = v\ then\ t\ else\ va\ w)\ (V, \$\in_r) \\
\qquad\quad) \\
\quad)
\end{array}$$

This should be understood as taking a formula and returning the formula which asserts the existence of the set with that extension, which differs from an instance of a comprehension axiom only in possibly having free variables. So we want to close this formula and then quantify over the stratified properties.

We now define the property of membership structures which consists in being closed under stratified comprehension. This has to have a variable name supplied as a parameter to make the definition type check, but the argument is ignored.

HOL Constant

$$\boldsymbol{StratCompClosed}: 'v \to ('s)PMS$$

$$\begin{array}{l}
\forall v\ ms \bullet\ StratCompClosed\ v\ ms = \\
\quad \forall sp \bullet\ StratProp\ sp \Rightarrow \forall v{:}'v \bullet\ foc\ (Comp\ v\ sp)\ ms
\end{array}$$

4.2 An Axiom of Stratified Comprehension

In this section we address the formulation of an axiom of stratified comprehension for NF and NFU.

This could be achived using the property *StratCompClosed* along the following lines:

$$\vdash StratCompClosed\ (Universe, \in_{nf})$$

However, if would be nice to have something which is a bit closer to the way an axiom of comprehension would be expected to look, along the lines:

$$\vdash \forall v\ p\bullet\ \mathit{Stratified}\ p \Rightarrow \exists s\bullet\ \forall x\bullet\ x \in_{nf} s \Leftrightarrow p\ v\ x$$

Where 'v' is a bundle of variables packaged together nas a function.

The property *Stratified* would have to be defined in the relevant context since it would have to make use of the correct membership relation. To make this possible we define here the closest support for this we can, viz. a notion of stratified property which is parameterised by the membership relationship. Assuming we are axiomatising a set theory as a new type so that the domain of the relationship us the whole type this gives us:

$$\vdash \forall va\ v\ p\bullet\ \mathit{Stratified}\ p \Rightarrow \exists s\bullet$$
$$\forall x\bullet\ x \in_{nf} s \Leftrightarrow p\ \$\in_{nf}\ (\lambda y\bullet\ \mathit{if}\ y = v\ \mathit{then}\ x\ \mathit{else}\ va\ y)$$

It is this which we now seek to define. *Stratified* here plays a role similar to that of StratProp but needs to have a rather different type, e.g.:
$\ulcorner : ('s \to 's \to BOOL) \to (('v \to 's) \to 's \to BOOL) \to BOOL \urcorner$
whereas *StratProp* has type: $\ulcorner : ('s, 'v)PPMS \to BOOL \urcorner$
which is similar to
$\ulcorner : ('v \to 's) \to (('s\ SET \times ('s \to 's \to BOOL)) \to BOOL) \to BOOL \urcorner$

So here goes:

HOL Constant

$$\boldsymbol{Stratified}: (('s \to 's \to BOOL) \to (STRING \to 's) \to 's\ SET) \to BOOL$$

$$\forall p\bullet\quad \mathit{Stratified}\ p$$
$$\Leftrightarrow$$
$$\mathit{StratProp}\ (\mathit{MkPPMS}\ \lambda va{:}\ \mathit{STRING} + \mathit{ONE} \to 's;\ (V,\ mr)\bullet$$
$$(va\ (\mathit{InR}\ \mathit{One})) \in p\ mr\ (va\ o\ \mathit{InL}))$$

5 BOOLEAN VALUED MEMBERSHIP STRUCTURES

Following Thomas Zech [1].

5.1 Filters

SML

```
open_theory "membership";
force_new_theory "fba";
set_merge_pcs["hol", "'savedthm_cs_∃_proof"];
```

HOL Constant

$$\boldsymbol{Filter}: 'a\ SET\ SET \to BOOL$$

$$\forall f \bullet \quad Filter\ f \Leftrightarrow \bigcup f \in f \wedge \neg\ \{\} \in f \wedge (\forall x\ y \bullet\ x \in f \wedge y \in f \Rightarrow x \cap y \in f) \wedge (\forall x\ y \bullet\ x \subseteq \bigcup f \wedge y \subseteq \bigcup f \wedge x \in f \wedge x \subseteq y \Rightarrow y \in f)$$

HOL Constant

$$\boldsymbol{Ideal}: 'a\ SET\ SET \to BOOL$$

$$\forall i \bullet \quad Ideal\ i \Leftrightarrow \neg \bigcup i \in i \wedge \{\} \in i \wedge (\forall x\ y \bullet\ x \in i \wedge y \in i \Rightarrow x \cup y \in i) \wedge (\forall x\ y \bullet\ x \subseteq \bigcup i \wedge y \subseteq \bigcup i \wedge x \in i \wedge y \subseteq x \Rightarrow y \in i)$$

5.2 Boolean Algebras

SML

```
open_theory "rbjmisc";
force_new_theory "ba";
new_parent "ordered_sets";
set_merge_pcs["hol", "'Z", "'savedthm_cs_∃_proof"];
```

HOL Labelled Product

BA

$$Car_{BA} :'a\ SET;$$
$$Sum_{BA} :'a \to 'a \to 'a;$$
$$Prod_{BA} :'a \to 'a \to 'a;$$
$$Comp_{BA} :'a \to 'a;$$
$$Bot_{BA} :'a;$$
$$Top_{BA} :'a$$

SML

```
declare_infix (310, "+B");
declare_infix (320, ".B");
```

HOL Constant

$$\boldsymbol{BooleanAlgebra} : 'a\ BA \to BOOL$$

$$\begin{array}{l}
\forall\ BA \bullet \\
\quad BooleanAlgebra\ BA \\
\Leftrightarrow \forall B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \bullet\ BA = MkBA\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \\
\quad \Rightarrow 0_B \in B \wedge 1_B \in B \\
\quad \wedge \forall\ u\ v\ w \bullet\ u \in B \wedge v \in B \wedge w \in B \\
\qquad \Rightarrow \quad u +_B v \in B \\
\qquad \wedge u ._B v \in B \\
\qquad \wedge \sim_B u \in B \\
\qquad \wedge u +_B v = v +_B u \\
\qquad \wedge u +_B (v +_B w) = (u +_B v) +_B w \\
\qquad \wedge u ._B v = v ._B u \\
\qquad \wedge u ._B (v ._B w) = (u ._B v) ._B w \\
\qquad \wedge u ._B (v +_B w) = (u ._B v) +_B (u ._B w) \\
\qquad \wedge u ._B (v +_B w) = (u ._B v) +_B (u ._B w) \\
\qquad \wedge u +_B (v ._B w) = (u +_B v) ._B (u +_B w) \\
\qquad \wedge u ._B (u +_B v) = u \\
\qquad \wedge u +_B (u ._B v) = u \\
\qquad \wedge u +_B (\sim_B u) = 1_B \\
\qquad \wedge u ._B (\sim_B u) = 0_B
\end{array}$$

SML

```
declare_infix(310, "−B");
declare_infix(300, "≤B");
```

SML

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

5.3 Filters and Ideals over Boolean Algebras

HOL Constant

$$\boldsymbol{Filter_{BA}} : 'a\ BA \to 'a\ SET \to BOOL$$

$$\begin{array}{l}
\forall BA\ f\ \bullet \\
\quad Filter_{BA}\ BA\ f \\
\quad \Leftrightarrow \\
\quad \forall B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \bullet \\
\quad (MkBA\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B) = BA \\
\quad \Rightarrow \\
\quad let\ x \leq_B y = x ._B (\sim_B y) = 0_B \\
\quad in\ f \subseteq B \\
\quad \wedge \quad 1_B \in f \wedge \neg\ 0_B \in f \\
\quad \wedge \quad (\forall x\ y \bullet\ x \in f \wedge y \in f \Rightarrow x ._B y \in f) \\
\quad \wedge \quad (\forall x\ y \bullet\ x \in f \wedge y \in B \wedge x \leq_B y \Rightarrow y \in f)
\end{array}$$

HOL Constant

$$\boldsymbol{Ideal_{BA}}: 'a\ BA \to 'a\ SET \to BOOL$$

$$\forall BA\ i \bullet$$
$$Ideal_{BA}\ BA\ i$$
$$\Leftrightarrow$$
$$\forall B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \bullet$$
$$(MkBA\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B) = BA$$
$$\Rightarrow$$
$$let\ \ x \leq_B y = x\ ._B\ (\sim_B\ y) = 0_B$$
$$in\ i \subseteq B$$
$$\wedge\ \ \neg\ 1_B \in i \wedge 0_B \in i$$
$$\wedge\ \ (\forall x\ y \bullet\ x \in i \wedge y \in i \Rightarrow x +_B y \in i)$$
$$\wedge\ \ (\forall x\ y \bullet\ y \in i \wedge x \in B \wedge x \leq_B y \Rightarrow x \in i)$$

5.4 Boolean Valued Interpretations

Here we define the rather more elaborate structures which we generalise the notion of a membership structure.

They do so in two distinct ways. Firstly, instead of allowing only classical relations as interpretations of the membership relation, they allow membership to be interpreted by functions into some boolean algebra. Secondly. they allow that equality be similarly liberalised. In a classical membership structure membership is not mentioned, and is assumed to be the standard equality, in which two elements of the domain are equal as sets if and only if they are in fact the same element.

SML

```
open_theory "ba";
force_new_theory "bvi";
```

HOL Labelled Product

BI

U_{BI} : $'a\ SET$;
BA_{BI} : $'b\ BA$;
Eq_{BI} : $'a \to 'a \to 'b$;
Mem_{BI} : $'a \to 'a \to 'b$

SML

```
declare_infix (300, "=B");
declare_infix (300, "∈B");
```

This should require that the boolean algebra be complete.

HOL Constant

$$\boldsymbol{BoolInterp} : ('a, 'b)\ BI \to BOOL$$

$$\begin{array}{l}
\forall\ i \bullet \\
\quad BoolInterp\ i \\
\Leftrightarrow \forall A\ x\ y\ z\ v\ w\ \$=_B\ \$\in_B\ (B, \$+_B, \$._B, \sim_B, 0_B, 1_B) \bullet \\
\qquad A = U_{BI}\ i \wedge MkBA\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B = BA_{BI}\ i \\
\quad \wedge\ \$=_B = Eq_{BI}\ i \wedge \$\in_B = Mem_{BI}\ i \\
\quad \wedge\ x \in A \wedge y \in A \wedge z \in A \wedge v \in A \wedge w \in A \\
\quad \Rightarrow \\
\quad let\ \ x \leq_B y = x\ ._B\ (\sim_B\ y) = 0_B \\
\quad in\ \ x =_B x = 1_B \\
\quad \wedge\ \ x =_B y = y =_B x \\
\quad \wedge\ \ (x =_B y)\ ._B\ \ (y =_B z) \leq_B x =_B z \\
\quad \wedge\ \ (x \in_B y)\ ._B\ \ (v =_B x)\ ._B\ \ (w =_B y) \leq_B v \in_B w
\end{array}$$

6 MEMBERSHIP FUNCTORS

One way of obtaining an interpretation of set theory is as follows:

1. chose some domain of set representatives
2. define a membership relation over the representatives

If the intended interpretation is not well-founded, the natural definition of the membership relation over the representatives may be recursive, and there will be a problem in establishing that the recursion is well-founded or in establishing by some other means that the functor encapsulating the recursive definition has a fixed point.

This approach to constructing interpretations of set theory is adopted in [3] using representatives which are definitions of properties in an infinitary set theory, but there are some general results in relation to functors over membership relations which are useful there but do not belong in that specific context.

We introduce a new theory for this material:

SML

```
open_theory "membership";
force_new_theory "memfunct";
force_new_pc "'memfunct";
merge_pcs ["'savedthm_cs_∃_proof"] "'memfunct";
set_merge_pcs ["hol", "'memfunct"];
```

7 The Theory membership

7.1 Parents

bin_rel ordered_sets rbjmisc

7.2 Children

memfunct fba

7.3 Constants

Ex	$((' a\ \mathbb{P}, ' a)\ VA, ' a\ MS)\ VA$
Co	$((' a, ' a\ \mathbb{P})\ VA, ' a\ MS)\ VA$
MS_ops	$(' a \times (' a, ' a \times ' a)\ VA \times ((' a, ' a)\ VA, ' a)\ VA \times (' a, ' a)\ VA \times (' a, ' a)\ VA \times ((' a, ' a\ \mathbb{P})\ VA, ' a)\ VA \times ((' a, (' a, ' a)\ VA)\ VA, ' a)\ VA, ' a\ MS)\ VA$
union	$((' a, ' a)\ VA, ' a\ MS)\ VA$
$\mathbf{Abs}_{x}$	$((' a, ' a\ \mathbb{P})\ VA, ' a\ XMS)\ VA$
$\mathbf{Snd}_{x}$	$((' a, ' a)\ VA, ' a\ XMS)\ VA$
$\mathbf{Fst}_{x}$	$((' a, ' a)\ VA, ' a\ XMS)\ VA$
$\mathbf{Op}_{x}$	$((' a, ' a \times ' a)\ VA, ' a\ XMS)\ VA$
$\varnothing_{x}$	$(' a, ' a\ XMS)\ VA$
$\mathbf{In}_{x}$	$(((BOOL, ' a)\ VA, ' a)\ VA, ' a\ XMS)\ VA$
$\mathbf{Car}_{x}$	$(' a\ \mathbb{P}, ' a\ XMS)\ VA$
MkXMS	$(((((((' a\ XMS, (' a, ' a\ \mathbb{P})\ VA)\ VA, (' a, ' a)\ VA)\ VA, (' a, ' a)\ VA)\ VA, (' a, ' a \times ' a)\ VA)\ VA, ' a)\ VA, ((BOOL, ' a)\ VA, ' a)\ VA)\ VA, ' a\ \mathbb{P})\ VA$
is_XMS	$(BOOL, ' a\ XMS)\ VA$
extensional	$(BOOL, ' a\ MS)\ VA$
weakly_extensional	$(BOOL, ' a\ MS)\ VA$
$\wedge_{r}$	$(((BOOL, ' a)\ VA, ' a)\ VA, ((BOOL, ' a)\ VA, ' a)\ VA\ \mathbb{P})\ VA$
ExtQuot	$(' a\ \mathbb{P}\ \mathbb{P}, ' a\ MS)\ VA$
ExtRel	$(' a\ \mathbb{P}\ MS, ' a\ MS)\ VA$
homomorph	$((BOOL, (' b, ' a)\ VA)\ VA, ' a\ MS \times ' b\ MS)\ VA$
embedding	$((BOOL, (' b, ' a)\ VA)\ VA, ' a\ MS \times ' b\ MS)\ VA$
$\$\mathbf{substructure\_of}$	$((BOOL, ' a\ MS)\ VA, ' a\ MS)\ VA$
$\$\mathbf{xsubstr}$	$((BOOL, ' a\ MS)\ VA, ' a\ MS)\ VA$
$\$\lhd_{m}$	$((' a\ MS, ' a\ MS)\ VA, (BOOL, ' a)\ VA)\ VA$
$\mathbf{ms\_}\varnothing$	$((BOOL, ' a)\ VA, ' a\ MS)\ VA$
$\mathbf{Ordinal}_{ms}$	$(BOOL, ' a\ MS)\ VA$

$\subseteq_{ms}$	$(((BOOL, 'a)\ VA, 'a)\ VA, 'a\ MS)\ VA$
TransitiveSet	
	$((BOOL, 'a)\ VA, 'a\ MS)\ VA$
set2rel	$(('a\ MS, 'a)\ VA, 'a\ MS)\ VA$
Ordinal	$((BOOL, 'a)\ VA, 'a\ MS)\ VA$
ordinals	$('a\ \mathbb{P}, 'a\ MS)\ VA$
$\varnothing$	$('a, 'a\ MS)\ VA$
$\boldsymbol{Succ_o}$	$(((BOOL, 'a)\ VA, 'a)\ VA, 'a\ MS)\ VA$
$\boldsymbol{SuccClosed_o}$	$(BOOL, 'a\ MS)\ VA$
$\boldsymbol{succ_o}$	$(('a, 'a)\ VA, 'a\ MS)\ VA$
$\boldsymbol{pred_o}$	$(('a, 'a)\ VA, 'a\ MS)\ VA$
$\boldsymbol{sum_o}$	$((('a, 'a)\ VA, 'a)\ VA, 'a\ MS)\ VA$
xms_ordinals	$('a\ \mathbb{P}, 'a\ XMS)\ VA$
ms_pair	$(((BOOL, 'a)\ VA, 'a \times 'a)\ VA, 'a\ MS)\ VA$
ms_unit	$(((BOOL, 'a)\ VA, 'a)\ VA, 'a\ MS)\ VA$
ms_wkp	$(((BOOL, 'a)\ VA, 'a \times 'a)\ VA, 'a\ MS)\ VA$
Vf	$(('a, ((BOOL, 'b)\ VA, 'b)\ VA \times 'a\ MS)\ VA,$ $('a, ((BOOL, 'b)\ VA, 'b)\ VA \times 'a\ MS)\ VA)\ VA$
ppms	$(((BOOL, 's\ MS)\ VA, ('s, 'v)\ VA)\ VA, ('s, 'v)\ PPMS)\ VA$
MkPPMS	$(('s, 'v)\ PPMS, ((BOOL, 's\ MS)\ VA, ('s, 'v)\ VA)\ VA)\ VA$
$\$\in_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, 'v)\ VA, 'v)\ VA$
$\$=_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, 'v)\ VA, 'v)\ VA$
$\$\wedge_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, ('s, 'v)\ PPMS)\ VA$
$\neg_{\boldsymbol{p}}$	$(('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA$
assign	$((((('s, 'v)\ VA, ('s, 'v)\ VA)\ VA, 'v)\ VA, 's)\ VA$
$\exists_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, 'v)\ VA$
fof	$(BOOL, ('s, 'v)\ PPMS)\ VA$
foc	$((BOOL, 's\ MS)\ VA, ('s, 'v)\ PPMS)\ VA$
fop	$((BOOL, (BOOL, 's\ MS)\ VA)\ VA, 'v)\ VA$
$\$\vee_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, ('s, 'v)\ PPMS)\ VA$
$\$\Rightarrow_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, ('s, 'v)\ PPMS)\ VA$
$\$\Leftrightarrow_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, ('s, 'v)\ PPMS)\ VA$
$\forall_{\boldsymbol{p}}$	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, 'v)\ VA$
Disjoint	$(BOOL, 'a\ \mathbb{P}\ LIST)\ VA$
SetMap	$(('b\ \mathbb{P}, 'a\ \mathbb{P})\ VA, ('b, 'a)\ VA)\ VA$
TaDom	$('v\ \mathbb{P}, 'v\ TA)\ VA$
WfTa	$(BOOL, 'v\ TA)\ VA$
$\$\lhd_{\boldsymbol{ta}}$	$(('v\ TA, 'v\ TA)\ VA, 'v\ \mathbb{P})\ VA$
Check_eq	$(((BOOL, 'v)\ VA, 'v)\ VA, 'v\ TA)\ VA$
Check_∈	$(((BOOL, 'v)\ VA, 'v)\ VA, 'v\ TA)\ VA$
wtfof	$((BOOL, ('s, 'v)\ PPMS)\ VA, 'v\ TA)\ VA$
StratProp	$(BOOL, ('s, 'v)\ PPMS)\ VA$
Comp	$((('s, 'v)\ PPMS, ('s, 'v)\ PPMS)\ VA, 'v)\ VA$
StratCompClosed	
	$((BOOL, 's\ MS)\ VA, 'v)\ VA$
Stratified	$(BOOL,$ $(('s\ \mathbb{P}, ('s, STRING)\ VA)\ VA, ((BOOL, 's)\ VA, 's)\ VA)$ $VA)\ VA$

7.4 Aliases

$\subseteq$	$\subseteq_{ms} : (((BOOL, 'a)\ VA, 'a)\ VA, 'a\ MS)\ VA$

7.5 Types

'1 ***XMS***
('1, '2) ***PPMS***

7.6 Type Abbreviations

'a ***MS***	*'a MS*
'a ***PMS***	*(BOOL, 'a MS) VA*
('s, 'v) ***VA***	*('s, 'v) VA*
'v ***TA***	*'v TA*

7.7 Fixity

Right Infix 290:
$\Leftrightarrow_{\boldsymbol{p}}$
Right Infix 295:
$\Rightarrow_{\boldsymbol{p}}$
Right Infix 298:
$\vee_{\boldsymbol{p}}$
Right Infix 300:
substructure_of ***xsubstr***
Right Infix 302:
$\wedge_{\boldsymbol{p}}$
Right Infix 305:
$=_{\boldsymbol{p}} \in_{\boldsymbol{a}} \in_{\boldsymbol{b}} \in_{\boldsymbol{p}} \in_{\boldsymbol{r}}$
Right Infix 310:
$(\!|)\ \cup_{\boldsymbol{m}} \lhd_{\boldsymbol{m}} \lhd_{\boldsymbol{ta}} \restriction_{\boldsymbol{m}}$

7.8 Definitions

Ex $\vdash \forall\ (X,\ \$\in_r)\ s$
$\bullet\ Ex\ (X,\ \$\in_r)\ s = \{y | s \in X \wedge y \in X \wedge y \in_r s\}$

Co $\vdash \forall\ (X,\ \$\in_r)\ s$
$\bullet\ (\exists\ x \bullet\ x \in X \wedge Ex\ (X,\ \$\in_r)\ x = s)$
$\Rightarrow Ex\ (X,\ \$\in_r)\ (Co\ (X,\ \$\in_r)\ s) = s$

MS_ops $\vdash$ *ConstSpec*
$(\lambda\ MS\_ops'$
$\bullet\ \forall\ X\ \$\in_a \varnothing_m\ Pair_m\ \$\cup_m \bigcup_m \mathbb{P}_m\ \$\restriction_m\ \$(\!|)$
$\bullet\ MS\_ops'\ (X,\ \$\in_a)$
$= (\varnothing_m,\ Pair_m,\ \$\cup_m,\ \bigcup_m,\ \mathbb{P}_m,\ \$\restriction_m,$
$\$(\!|))$
$\Rightarrow (\forall\ ys\ y\ x$
$\bullet\ \varnothing_m = Co\ (X,\ \$\in_a)\ \{\}$
$\wedge\ Pair_m\ (x,\ y) = Co\ (X,\ \$\in_a)\ \{x;\ y\}$
$\wedge\ x \cup_m y$
$= Co$
$(X,\ \$\in_a)$
$(Ex\ (X,\ \$\in_a)\ x \cup Ex\ (X,\ \$\in_a)\ y)$
$\wedge \bigcup_m x$

$$
\begin{array}{l}
\quad = Co \\
\quad\quad (X, \$\in_a) \\
\quad\quad (\bigcup \\
\quad\quad\quad \{z \\
\quad\quad\quad\quad | \exists\ y \\
\quad\quad\quad\quad\quad \bullet\ y \in X \\
\quad\quad\quad\quad\quad\quad \wedge\ y \in_a x \\
\quad\quad\quad\quad\quad\quad \wedge\ z = Ex\ (X, \$\in_a)\ y\}) \\
\wedge\ \mathbb{P}_m\ x \\
\quad = Co \\
\quad\quad (X, \$\in_a) \\
\quad\quad \{y \\
\quad\quad\quad | Ex\ (X, \$\in_a)\ y \\
\quad\quad\quad\quad \in \mathbb{P}\ (Ex\ (X, \$\in_a)\ x)\} \\
\wedge\ x \upharpoonright_m ys \\
\quad = Co \\
\quad\quad (X, \$\in_a) \\
\quad\quad \{u | u \in X \wedge u \in_a x \wedge u \in ys\}))
\end{array}
$$

MS_ops

union $\vdash \forall\ (X, \$\in_a)\ x$

$$
\begin{array}{l}
\quad \bullet\ union\ (X, \$\in_a)\ x \\
\quad\quad = Co \\
\quad\quad\quad (X, \$\in_a) \\
\quad\quad\quad (\bigcup \\
\quad\quad\quad\quad \{z \\
\quad\quad\quad\quad\quad | \exists\ y \\
\quad\quad\quad\quad\quad\quad \bullet\ y \in X \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ y \in_a x \\
\quad\quad\quad\quad\quad\quad\quad \wedge\ z = Ex\ (X, \$\in_a)\ y\})
\end{array}
$$

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

MkXMS

Car$_x$

In$_x$

$\varnothing_{\boldsymbol{x}}$

Op$_x$

Fst$_x$

Snd$_x$

Abs$_x$ $\vdash \forall\ t\ x1\ x2\ x3\ x4\ x5\ x6\ x7$

$$
\begin{array}{l}
\quad \bullet\ Car_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x1 \\
\quad\quad \wedge\ In_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x2 \\
\quad\quad \wedge\ \varnothing_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x3 \\
\quad\quad \wedge\ Op_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x4 \\
\quad\quad \wedge\ Fst_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x5 \\
\quad\quad \wedge\ Snd_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x6 \\
\quad\quad \wedge\ Abs_x\ (MkXMS\ x1\ x2\ x3\ x4\ x5\ x6\ x7) = x7 \\
\quad\quad \wedge\ MkXMS \\
\quad\quad\quad (Car_x\ t) \\
\quad\quad\quad (In_x\ t) \\
\quad\quad\quad (\varnothing_x\ t) \\
\quad\quad\quad (Op_x\ t) \\
\quad\quad\quad (Fst_x\ t)
\end{array}
$$

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad\quad\quad (Snd_x\ t) \\
\quad\quad\quad\quad\quad\quad\quad\quad (Abs_x\ t) \\
\quad\quad\quad\quad\quad\quad\quad\quad = t \\
\textbf{\textit{is\_XMS}} \quad\quad\quad\quad\quad\quad \vdash \forall\ xms \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ is\_XMS\ xms \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\varnothing_x\ xms \in Car_x\ xms \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge \neg\ (\exists\ s \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ s \in Car_x\ xms \wedge In_x\ xms\ s\ (\varnothing_x\ xms))) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge (\forall\ l\ r \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ l \in Car_x\ xms \wedge r \in Car_x\ xms \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow Op_x\ xms\ (l,\ r) \in Car_x\ xms \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge Fst_x\ xms\ (Op_x\ xms\ (l,\ r)) = l \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge Snd_x\ xms\ (Op_x\ xms\ (l,\ r)) = r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge (\forall\ s \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ (\exists\ t\bullet \forall\ u\bullet\ u \in s \Leftrightarrow In_x\ xms\ u\ t) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow (\forall\ u\bullet\ u \in s \Leftrightarrow In_x\ xms\ u\ (Abs_x\ xms\ s))) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ WellFounded\ (Car_x\ xms,\ In_x\ xms) \\
\textbf{\textit{extensional}} \quad\quad\quad\quad\quad\quad \vdash \forall\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ extensional\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\forall\ s\ t \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ s \in X \wedge t \in X \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow (s = t \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\forall\ u\bullet\ u \in X \Rightarrow (u \in_r s \Leftrightarrow u \in_r t)))) \\
\textbf{\textit{weakly\_extensional}} \\
\quad\quad\quad\quad\quad\quad\quad\quad \vdash \forall\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ weakly\_extensional\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\forall\ s\ t \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ s \in X \wedge t \in X \wedge (\exists\ v\bullet\ v \in X \wedge v \in_r s) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow (s = t \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\forall\ u\bullet\ u \in X \Rightarrow (u \in_r s \Leftrightarrow u \in_r t)))) \\
\wedge_{\boldsymbol{r}} \quad\quad\quad\quad\quad\quad \vdash \forall\ sr\bullet \wedge_r\ sr = (\lambda\ x\ y\bullet \forall\ z\bullet\ z \in sr \Rightarrow z\ x\ y) \\
\textbf{\textit{ExtQuot}} \quad\quad\quad\quad\quad\quad \vdash \forall\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ ExtQuot\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = X \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad / \wedge_r \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \{r \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad |Equiv\ (X,\ r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge (\forall\ x\ y \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ x \in X \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ y \in X \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge ((\forall\ z\bullet\ z \in_r x \Leftrightarrow z \in_r y) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Rightarrow r\ x\ y))\} \\
\textbf{\textit{ExtRel}} \quad\quad\quad\quad\quad\quad \vdash \forall\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ ExtRel\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = (let\ Y = ExtQuot\ (X,\ \$\in_r) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad in\ (Y, \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (\lambda\ l\ r\bullet \exists\ u\ v\bullet\ u \in l \wedge v \in r \wedge u \in_r v))) \\
\textbf{\textit{homomorph}} \quad\quad\quad\quad\quad\quad \vdash \forall\ (X,\ \$\in_a)\ (Y,\ \$\in_b)\ f \\
\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ homomorph\ ((X,\ \$\in_a),\ Y,\ \$\in_b)\ f \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow (\forall\ s\ t \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ s \in X \wedge t \in X \wedge s \in_a t
\end{array}
$$

$\Rightarrow f\ s \in Y \wedge f\ t \in Y \wedge f\ s \in_b f\ t)$

embedding $\vdash \forall\ (X, \$\in_a)\ (Y, \$\in_b)\ f$
$\bullet$ *embedding* $((X, \$\in_a),\ Y, \$\in_b)\ f$
$\Leftrightarrow$ *OneOne* f
$\wedge\ (\forall\ s\ t$
$\bullet\ s \in X \wedge t \in X$
$\Rightarrow f\ s \in Y$
$\wedge f\ t \in Y$
$\wedge\ (s \in_a t \Leftrightarrow f\ s \in_b f\ t))$

substructure_of $\vdash \forall\ (X, \$\in_a)\ (Y, \$\in_b)$
$\bullet\ (X, \$\in_a)$ *substructure_of* $(Y, \$\in_b)$
$\Leftrightarrow X \subseteq Y$
$\wedge\ (\forall\ s\ t\bullet\ s \in X \wedge t \in X \Rightarrow (s \in_a t \Leftrightarrow s \in_b t))$

xsubstr $\vdash \forall\ (X, \$\in_a)\ (Y, \$\in_b)$
$\bullet\ (X, \$\in_a)\ xsubstr\ (Y, \$\in_b)$
$\Leftrightarrow X \subseteq Y$
$\wedge\ (\forall\ s\ t\bullet\ t \in X \wedge s \in Y \Rightarrow (s \in_a t \Leftrightarrow s \in_b t))$

$\lhd_{\boldsymbol{m}}$ $\vdash \forall\ P\ (X, \$\in_a)\bullet\ P \lhd_m (X, \$\in_a) = (X \cap \{x|P\ x\}, \$\in_a)$

ms_∅ $\vdash \forall\ (X, \$\in_a)\ s$
$\bullet\ ms\_\varnothing\ (X, \$\in_a)\ s \Leftrightarrow s \in X \wedge (\forall\ z\bullet\ z \in X \Rightarrow \neg\ z \in_a s)$

Ordinal$_{ms}$ $\vdash \forall\ ms$
$\bullet$ *Ordinal*$_{ms}$ *ms*
$\Leftrightarrow$ *LinearOrder ms* $\wedge$ *Trans ms* $\wedge$ *WellFounded ms*

$\subseteq_{\boldsymbol{ms}}$ $\vdash \forall\ (X, \$\in_r)\ a\ b$
$\bullet \subseteq (X, \$\in_r)\ a\ b$
$\Leftrightarrow (\forall\ x\bullet\ x \in X \wedge x \in_r a \Rightarrow x \in_r b)$

TransitiveSet $\vdash \forall\ (X, \$\in_r)\ a$
$\bullet$ *TransitiveSet* $(X, \$\in_r)\ a$
$\Leftrightarrow (\forall\ x\bullet\ x \in X \wedge x \in_r a \Rightarrow \subseteq (X, \$\in_r)\ x\ a)$

set2rel $\vdash \forall\ (X, \$\in_r)\ a$
$\bullet$ *set2rel* $(X, \$\in_r)\ a = (\{x|x \in_r a \wedge x \in X\}, \$\in_r)$

Ordinal $\vdash \forall\ ms\ s$
$\bullet$ *Ordinal ms s*
$\Leftrightarrow$ *TransitiveSet ms s*
$\wedge$ *WellOrdering* (*set2rel ms s*)

ordinals $\vdash \forall\ ms\bullet\ ordinals\ ms = \{s|Ordinal\ ms\ s\}$

$\varnothing$ $\vdash \forall\ (X, \$\in_a)\bullet\ \varnothing\ (X, \$\in_a) = Co\ (X, \$\in_a)\ \{\}$

Succ$_o$ $\vdash \forall\ ms\ a\ b$
$\bullet\ Succ_o\ ms\ a\ b \Leftrightarrow Ex\ ms\ b = \{x|x = a \vee x \in Ex\ ms\ a\}$

SuccClosed$_o$ $\vdash \forall\ ms$
$\bullet$ *SuccClosed*$_o$ *ms*
$\Leftrightarrow (\forall\ x\bullet\ Ordinal\ ms\ x \Rightarrow (\exists\ y\bullet\ Succ_o\ ms\ x\ y))$

succ$_o$ $\vdash \forall\ (X, \$\in_a)\ x$
$\bullet\ succ_o\ (X, \$\in_a)\ x$
$= Co\ (X, \$\in_a)\ \{y|y \in X \wedge (y = x \vee y \in_a x)\}$

pred$_o$ $\vdash$ *ConstSpec*
$(\lambda\ pred_o{}'$
$\bullet\ \forall\ ms\ x$
$\bullet\ Ordinal\ ms\ x \Rightarrow pred_o{}'\ ms\ (succ_o\ ms\ x) = x)$

$pred_o$

sum_o $\vdash$ *ConstSpec*
$(\lambda\ sum_o'$
$\bullet\ \forall\ ms\ x\ y$
$\bullet\ \{x;\ y\} \subseteq ordinals\ ms$
$\Rightarrow sum_o'\ ms\ x\ (\varnothing\ ms) = x$
$\wedge\ sum_o'\ ms\ x\ (succ_o\ ms\ y)$
$= succ_o\ ms\ (sum_o'\ ms\ x\ y))$
sum_o

xms_ordinals $\vdash \forall\ xms$
$\bullet\ xms\_ordinals\ xms$
$= \{x$
$|x \in Car_x\ xms \wedge Ordinal\ (Car_x\ xms,\ In_x\ xms)\ x\}$

ms_pair $\vdash \forall\ (X,\ \$\in_a)\ (l,\ r)\ p$
$\bullet\ ms\_pair\ (X,\ \$\in_a)\ (l,\ r)\ p$
$\Leftrightarrow p \in X$
$\wedge\ (\forall\ z\bullet\ z \in X \Rightarrow (z \in_a p \Leftrightarrow z = l \vee z = r))$

ms_unit $\vdash \forall\ ms\ s\ u\bullet\ ms\_unit\ ms\ s\ u \Leftrightarrow ms\_pair\ ms\ (s,\ s)\ u$

ms_wkp $\vdash \forall\ ms\ (l,\ r)\ p$
$\bullet\ ms\_wkp\ ms\ (l,\ r)\ p$
$\Leftrightarrow (\exists\ s\ t$
$\bullet\ ms\_unit\ ms\ l\ s$
$\wedge\ ms\_pair\ ms\ (l,\ r)\ t$
$\wedge\ ms\_pair\ ms\ (s,\ t)\ p)$

Vf $\vdash \forall\ f$
$\bullet\ Vf\ f$
$= (\lambda\ (\$<<,\ ms)$
$\bullet\ (let\ (\varnothing_m,\ Pair_m,\ \$\cup_m,\ \bigcup_m,\ \mathbb{P}_m,\ \$\restriction_m,\ \$(\!|))$
$= MS\_ops\ ms$
$in\ Co$
ms
$\{z$
$|z \in Fst\ ms$
$\wedge\ (\exists\ v$
$\bullet\ Ex\ ms\ z$
$\subseteq Ex$
ms
$(f$
$((\lambda\ x\ y\bullet\ x << y \wedge y << v),$
$ms)))\})$

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

MkPPMS

ppms $\vdash \forall\ t\ x1\bullet\ ppms\ (MkPPMS\ x1) = x1 \wedge MkPPMS\ (ppms\ t) = t$

$\in_p$ $\vdash \forall\ s\ t$
$\bullet\ s \in_p t = MkPPMS\ (\lambda\ va\ (X,\ \$\in_r)\bullet\ va\ s \in_r va\ t)$

$=_p$ $\vdash \forall\ s\ t\bullet\ s =_p t = MkPPMS\ (\lambda\ va\ (X,\ \$\in_r)\bullet\ va\ s = va\ t)$

$\wedge_p$ $\vdash \forall\ l\ r$
$\bullet\ l \wedge_p r$
$= MkPPMS$
$(\lambda\ va\ (X,\ \$\in_r)$
$\bullet\ ppms\ l\ va\ (X,\ \$\in_r) \wedge ppms\ r\ va\ (X,\ \$\in_r))$

$\neg_p$ $\vdash \forall\ f$
$\bullet\ \neg_p\ f$
$= MkPPMS\ (\lambda\ va\ (X, \$\in_r)\bullet\ \neg\ ppms\ f\ va\ (X, \$\in_r))$

assign $\vdash \forall\ a\ s\ va\ t$
$\bullet\ assign\ a\ s\ va\ t = (if\ t = s\ then\ a\ else\ va\ t)$

$\exists_p$ $\vdash \forall\ s\ f$
$\bullet\ \exists_p\ s\ f$
$= MkPPMS$
$(\lambda\ va\ (X, \$\in_r)$
$\bullet\ \exists\ a$
$\bullet\ a \in X \wedge ppms\ f\ (assign\ a\ s\ va)\ (X, \$\in_r))$

fof $\vdash \forall\ ppms1$
$\bullet\ fof\ ppms1$
$\Leftrightarrow (\forall\ pppms$
$\bullet\ (\forall\ s1\ s2\ p1\ p2$
$\bullet\ pppms\ (s1 \in_p s2)$
$\wedge\ pppms\ (s1 =_p s2)$
$\wedge\ (pppms\ p1 \wedge pppms\ p2$
$\Rightarrow pppms\ (p1 \wedge_p p2)$
$\wedge\ pppms\ (\neg_p\ p1)$
$\wedge\ pppms\ (\exists_p\ s1\ p1)))$
$\Rightarrow pppms\ ppms1)$

foc $\vdash \forall\ ppms1\ (X, \in_m)$
$\bullet\ foc\ ppms1\ (X, \in_m)$
$\Leftrightarrow (\forall\ va$
$\bullet\ X = \{\} \vee (\forall\ v\bullet\ va\ v \in X)$
$\Rightarrow ppms\ ppms1\ va\ (X, \in_m))$

fop $\vdash \forall\ pmr\ v\bullet\ fop\ v\ pmr \Leftrightarrow (\exists\ f\bullet\ fof\ f \wedge pmr = foc\ f)$

$\vee_p$ $\vdash \forall\ l\ r\bullet\ l \vee_p r = \neg_p\ (\neg_p\ l \wedge_p \neg_p\ r)$

$\Rightarrow_p$ $\vdash \forall\ l\ r\bullet\ l \Rightarrow_p r = r \vee_p \neg_p\ l$

$\Leftrightarrow_p$ $\vdash \forall\ l\ r\bullet\ l \Leftrightarrow_p r = (l \Rightarrow_p r) \wedge_p (r \Rightarrow_p l)$

$\forall_p$ $\vdash \forall\ s\ p\bullet\ \forall_p\ s\ p = \neg_p\ (\exists_p\ s\ (\neg_p\ p))$

Disjoint $\vdash (Disjoint\ [] \Leftrightarrow T)$
$\wedge\ (\forall\ h\ t$
$\bullet\ Disjoint\ (Cons\ h\ t)$
$\Leftrightarrow h \cap ListUnion\ t = \{\} \wedge Disjoint\ t)$

SetMap $\vdash \forall\ f\ s\bullet\ SetMap\ f\ s = \{x | \exists\ y\bullet\ y \in s \wedge x = f\ y\}$

TaDom $\vdash \forall\ ta\bullet\ TaDom\ ta = ListUnion\ (Map\ ListUnion\ ta)$

WfTa $\vdash \forall\ ta$
$\bullet\ WfTa\ ta$
$\Leftrightarrow \forall_L\ (Map\ Disjoint\ ta)$
$\wedge\ Disjoint\ (Map\ ListUnion\ ta)$

$\lhd_{ta}$ $\vdash \forall\ vs\ ta\bullet\ vs \lhd_{ta} ta = Map\ (\lambda\ l\bullet\ Map\ (\$\cap\ vs)\ l)\ ta$

Check_eq $\vdash \forall\ ta\ a\ b$
$\bullet\ Check\_eq\ ta\ a\ b \Leftrightarrow \{a; b\} \lhd_{ta} ta = [[\{a; b\}]]$

Check_∈ $\vdash \forall\ ta\ a\ b$
$\bullet\ Check\_\in\ ta\ a\ b \Leftrightarrow \{a; b\} \lhd_{ta} ta = [[\{a\}; \{b\}]]$

wtfof $\vdash \forall\ ppms1\ ta$
$\bullet\ wtfof\ ta\ ppms1$
$\Leftrightarrow (\forall\ pppms$
$\bullet\ (\forall\ s1\ s2\ p1\ p2$

$$
\begin{array}{l}
\quad \bullet\ (Check\_\in\ ta\ s1\ s2 \Rightarrow pppms\ (s1 \in_p s2)) \\
\quad\quad \wedge\ (Check\_eq\ ta\ s1\ s2 \Rightarrow pppms\ (s1 =_p s2)) \\
\quad\quad \wedge\ (pppms\ p1 \wedge pppms\ p2 \\
\quad\quad\quad \Rightarrow pppms\ (\neg_p\ p1) \\
\quad\quad\quad\quad \wedge\ pppms\ (\exists_p\ s1\ p1) \\
\quad\quad\quad\quad \wedge\ pppms\ (p1 \wedge_p p2))) \\
\quad\quad \Rightarrow pppms\ ppms1)
\end{array}
$$

StratProp $\vdash \forall\ ppms1$

$$
\begin{array}{l}
\quad \bullet\ StratProp\ ppms1 \Leftrightarrow (\exists\ ta \bullet\ WfTa\ ta \wedge wtfof\ ta\ ppms1)
\end{array}
$$

Comp $\vdash \forall\ v\ ppms1$

$$
\begin{array}{l}
\quad \bullet\ Comp\ v\ ppms1 \\
\quad\quad = MkPPMS \\
\quad\quad\quad (\lambda\ va\ (V, \$\in_r) \\
\quad\quad\quad\quad \bullet \exists\ s \\
\quad\quad\quad\quad\quad \bullet\ s \in V \\
\quad\quad\quad\quad\quad\quad \wedge\ (\forall\ t \\
\quad\quad\quad\quad\quad\quad \bullet\ t \in V \\
\quad\quad\quad\quad\quad\quad\quad \Rightarrow (t \in_r s \\
\quad\quad\quad\quad\quad\quad\quad\quad \Leftrightarrow ppms \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad ppms1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad (\lambda\ w \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \bullet\ if\ w = v\ then\ t\ else\ va\ w) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad (V, \$\in_r))))
\end{array}
$$

StratCompClosed

$\vdash \forall\ v\ ms$

$$
\begin{array}{l}
\quad \bullet\ StratCompClosed\ v\ ms \\
\quad\quad \Leftrightarrow (\forall\ sp \\
\quad\quad \bullet\ StratProp\ sp \Rightarrow (\forall\ v \bullet\ foc\ (Comp\ v\ sp)\ ms))
\end{array}
$$

Stratified $\vdash \forall\ p$

$$
\begin{array}{l}
\quad \bullet\ Stratified\ p \\
\quad\quad \Leftrightarrow StratProp \\
\quad\quad\quad (MkPPMS \\
\quad\quad\quad\quad (\lambda\ va\ (V, mr) \\
\quad\quad\quad\quad\quad \bullet\ va\ (InR\ One) \in p\ mr\ (va\ o\ InL)))
\end{array}
$$

7.9 Theorems

extensional_thm

$\vdash \forall\ ms$

$$
\begin{array}{l}
\quad \bullet\ extensional\ ms \\
\quad\quad \Leftrightarrow (\forall\ s\ t \\
\quad\quad \bullet\ s \in Fst\ ms \wedge t \in Fst\ ms \\
\quad\quad\quad \Rightarrow (s = t \\
\quad\quad\quad\quad \Leftrightarrow (\forall\ u \\
\quad\quad\quad\quad \bullet\ u \in Fst\ ms \Rightarrow (Snd\ ms\ u\ s \Leftrightarrow Snd\ ms\ u\ t))))
\end{array}
$$

foc_thm $\vdash \forall\ ppms1\ ms$

$$
\begin{array}{l}
\quad \bullet\ foc\ ppms1\ ms \\
\quad\quad \Leftrightarrow (\forall\ va \\
\quad\quad \bullet\ Fst\ ms = \{\} \vee (\forall\ v \bullet\ va\ v \in Fst\ ms) \\
\quad\quad\quad \Rightarrow ppms\ ppms1\ va\ ms)
\end{array}
$$

fof_base_clauses

$\vdash \forall\ s1\ s2 \bullet\ fof\ (s1 =_p s2) \wedge fof\ (s1 \in_p s2)$

fof_step_clauses

$\vdash \forall\ s\ s1\ p1\ p2$
$\bullet\ (fof\ p1$
$\Rightarrow fof\ (\neg_p\ p1)$
$\wedge\ fof\ (\exists_p\ s\ p1)$
$\wedge\ fof\ (\forall_p\ s\ p1))$
$\wedge\ (fof\ p1 \wedge fof\ p2$
$\Rightarrow fof\ (p1 \wedge_p p2)$
$\wedge\ fof\ (p1 \vee_p p2)$
$\wedge\ fof\ (p1 \Rightarrow_p p2)$
$\wedge\ fof\ (p1 \Leftrightarrow_p p2))$

fop_extensional_thm

$\vdash fop\ ""\ extensional$

SetMap_display

$\vdash \forall\ f$
$\bullet\ SetMap\ f\ \{\} = \{\}$
$\wedge\ (\forall\ s\ e$
$\bullet\ SetMap\ f\ (Insert\ e\ s)$
$= Insert\ (f\ e)\ (SetMap\ f\ s))$

Disp_∩_thm $\vdash \forall\ x\ s\ t \bullet\ Insert\ x\ s \cap t = \{x\} \cap t \cup s \cap t$

Disp_∩_thm2 $\vdash \forall\ x\ y\ v\ w$
$\bullet\ \{x\} \cap Insert\ x\ y = \{x\}$
$\wedge\ Insert\ x\ y \cap Insert\ x\ w = \{x\} \cup y \cap w$

8 The Theory ba

8.1 Parents

ordered_sets *rbjmisc*

8.2 Children

bvi

8.3 Constants

$\boldsymbol{Top_{BA}}$ $'a\ BA \to 'a$

$\boldsymbol{Bot_{BA}}$ $'a\ BA \to 'a$

$\boldsymbol{Comp_{BA}}$ $'a\ BA \to 'a \to 'a$

$\boldsymbol{Prod_{BA}}$ $'a\ BA \to 'a \to 'a \to 'a$

$\boldsymbol{Sum_{BA}}$ $'a\ BA \to 'a \to 'a \to 'a$

$\boldsymbol{Car_{BA}}$ $'a\ BA \to 'a\ \mathbb{P}$

$\boldsymbol{MkBA}$ $'a\ \mathbb{P}$
$\to ('a \to 'a \to 'a)$
$\to ('a \to 'a \to 'a)$
$\to ('a \to 'a)$
$\to 'a$
$\to 'a$
$\to 'a\ BA$

$\boldsymbol{BooleanAlgebra}$
$'a\ BA \to BOOL$

$\boldsymbol{Filter_{BA}}$ $'a\ BA \to 'a\ \mathbb{P} \to BOOL$

$\boldsymbol{Ideal_{BA}}$ $'a\ BA \to 'a\ \mathbb{P} \to BOOL$

8.4 Types

$'1\ \boldsymbol{BA}$

8.5 Fixity

Right Infix 300:
$\leq_{\boldsymbol{B}}$

Right Infix 310:
$+_{\boldsymbol{B}} -_{\boldsymbol{B}}$

Right Infix 320:
$\cdot_{\boldsymbol{B}}$

8.6 Definitions

$$\vdash \exists\, f \bullet\ \mathit{TypeDefn}\ (\lambda\, x \bullet\ T)\ f$$

BA
MkBA
Car$_{BA}$
Sum$_{BA}$
Prod$_{BA}$
Comp$_{BA}$
Bot$_{BA}$
Top$_{BA}$

$$\begin{array}{l}
\vdash \forall\ t\ x1\ x2\ x3\ x4\ x5\ x6 \\
\quad \bullet\ \mathit{Car}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x1 \\
\qquad \wedge\ \mathit{Sum}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x2 \\
\qquad \wedge\ \mathit{Prod}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x3 \\
\qquad \wedge\ \mathit{Comp}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x4 \\
\qquad \wedge\ \mathit{Bot}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x5 \\
\qquad \wedge\ \mathit{Top}_{BA}\ (\mathit{MkBA}\ x1\ x2\ x3\ x4\ x5\ x6) = x6 \\
\qquad \wedge\ \mathit{MkBA} \\
\qquad\quad (\mathit{Car}_{BA}\ t) \\
\qquad\quad (\mathit{Sum}_{BA}\ t) \\
\qquad\quad (\mathit{Prod}_{BA}\ t) \\
\qquad\quad (\mathit{Comp}_{BA}\ t) \\
\qquad\quad (\mathit{Bot}_{BA}\ t) \\
\qquad\quad (\mathit{Top}_{BA}\ t) \\
\qquad = t
\end{array}$$

BooleanAlgebra

$$\begin{array}{l}
\vdash \forall\ BA \\
\quad \bullet\ \mathit{BooleanAlgebra}\ BA \\
\qquad \Leftrightarrow (\forall\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \\
\qquad\quad \bullet\ BA = \mathit{MkBA}\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B \\
\qquad\qquad \Rightarrow 0_B \in B \\
\qquad\qquad\quad \wedge\ 1_B \in B \\
\qquad\qquad\quad \wedge\ (\forall\ u\ v\ w \\
\qquad\qquad\quad\quad \bullet\ u \in B \wedge v \in B \wedge w \in B \\
\qquad\qquad\quad\quad\quad \Rightarrow u +_B v \in B \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B v \in B \\
\qquad\qquad\quad\quad\quad\quad \wedge\ \sim_B u \in B \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u +_B v = v +_B u \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u +_B v +_B w = (u +_B v) +_B w \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B v = v ._B u \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B v ._B w = (u ._B v) ._B w \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B (v +_B w) \\
\qquad\qquad\quad\quad\quad\quad\quad = u ._B v +_B u ._B w \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B (v +_B w) \\
\qquad\qquad\quad\quad\quad\quad\quad = u ._B v +_B u ._B w \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u +_B v ._B w \\
\qquad\qquad\quad\quad\quad\quad\quad = (u +_B v) ._B (u +_B w) \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B (u +_B v) = u \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u +_B u ._B v = u \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u +_B \sim_B u = 1_B \\
\qquad\qquad\quad\quad\quad\quad \wedge\ u ._B \sim_B u = 0_B))
\end{array}$$

Filter$_{BA}$

$$\begin{array}{l}
\vdash \forall\ BA\ f \\
\quad \bullet\ \mathit{Filter}_{BA}\ BA\ f
\end{array}$$

$\Leftrightarrow (\forall\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$
$\bullet\ MkBA\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B = BA$
$\Rightarrow (let\ x \leq_B y \Leftrightarrow x\ .{}_B\ \sim_B\ y = 0_B$
$in\ f \subseteq B$
$\wedge\ 1_B \in f$
$\wedge\ \neg\ 0_B \in f$
$\wedge\ (\forall\ x\ y \bullet\ x \in f \wedge y \in f \Rightarrow x\ .{}_B\ y \in f)$
$\wedge\ (\forall\ x\ y$
$\bullet\ x \in f \wedge y \in B \wedge x \leq_B y \Rightarrow y \in f)))$

$Ideal_{BA}$ $\vdash \forall\ BA\ i$
$\bullet\ Ideal_{BA}\ BA\ i$
$\Leftrightarrow (\forall\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$
$\bullet\ MkBA\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B = BA$
$\Rightarrow (let\ x \leq_B y \Leftrightarrow x\ .{}_B\ \sim_B\ y = 0_B$
$in\ i \subseteq B$
$\wedge\ \neg\ 1_B \in i$
$\wedge\ 0_B \in i$
$\wedge\ (\forall\ x\ y \bullet\ x \in i \wedge y \in i \Rightarrow x +_B y \in i)$
$\wedge\ (\forall\ x\ y$
$\bullet\ y \in i \wedge x \in B \wedge x \leq_B y \Rightarrow x \in i)))$

8.7 Theorems

BA_fc_lem1 $\vdash \forall\ BA$
$\bullet\ BooleanAlgebra\ BA$
$\Rightarrow (\forall\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$
$\bullet\ BA = MkBA\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$
$\Rightarrow\ 0_B \in B$
$\wedge\ 1_B \in B$
$\wedge\ (\forall\ u\ v\ w$
$\bullet\ u \in B \wedge v \in B \wedge w \in B$
$\Rightarrow\ u +_B v \in B$
$\wedge\ u\ .{}_B\ v \in B$
$\wedge\ \sim_B u \in B$
$\wedge\ u +_B v = v +_B u$
$\wedge\ u +_B v +_B w = (u +_B v) +_B w$
$\wedge\ u\ .{}_B\ v = v\ .{}_B\ u$
$\wedge\ u\ .{}_B\ v\ .{}_B\ w = (u\ .{}_B\ v)\ .{}_B\ w$
$\wedge\ u\ .{}_B\ (v +_B w)$
$= u\ .{}_B\ v +_B u\ .{}_B\ w$
$\wedge\ u\ .{}_B\ (v +_B w)$
$= u\ .{}_B\ v +_B u\ .{}_B\ w$
$\wedge\ u +_B v\ .{}_B\ w$
$= (u +_B v)\ .{}_B\ (u +_B w)$
$\wedge\ u\ .{}_B\ (u +_B v) = u$
$\wedge\ u +_B u\ .{}_B\ v = u$
$\wedge\ u +_B \sim_B u = 1_B$
$\wedge\ u\ .{}_B\ \sim_B u = 0_B))$

BA_fc_lem2 $\vdash \forall\ BA\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$
$\bullet\ BooleanAlgebra\ BA$
$\wedge\ BA = MkBA\ B\ \$+_B\ \$.{}_B\ \sim_B\ 0_B\ 1_B$

$$
\begin{array}{l}
\Rightarrow 0_B \in B \\
\quad \wedge 1_B \in B \\
\quad \wedge (\forall\ u\ v\ w \\
\quad \bullet\ u \in B \wedge v \in B \wedge w \in B \\
\qquad \Rightarrow u +_B v \in B \\
\qquad\quad \wedge u \cdot_B v \in B \\
\qquad\quad \wedge \sim_B u \in B \\
\qquad\quad \wedge u +_B v = v +_B u \\
\qquad\quad \wedge u +_B v +_B w = (u +_B v) +_B w \\
\qquad\quad \wedge u \cdot_B v = v \cdot_B u \\
\qquad\quad \wedge u \cdot_B v \cdot_B w = (u \cdot_B v) \cdot_B w \\
\qquad\quad \wedge u \cdot_B (v +_B w) = u \cdot_B v +_B u \cdot_B w \\
\qquad\quad \wedge u \cdot_B (v +_B w) = u \cdot_B v +_B u \cdot_B w \\
\qquad\quad \wedge u +_B v \cdot_B w = (u +_B v) \cdot_B (u +_B w) \\
\qquad\quad \wedge u \cdot_B (u +_B v) = u \\
\qquad\quad \wedge u +_B u \cdot_B v = u \\
\qquad\quad \wedge u +_B \sim_B u = 1_B \\
\qquad\quad \wedge u \cdot_B \sim_B u = 0_B)
\end{array}
$$

BA_fc_lem3 $\vdash$ *BooleanAlgebra BA* $\wedge$ *BA* = *MkBA B* \$+$_B$ \$.$_B$ $\sim_B$ 0_B 1_B

$$
\begin{array}{l}
\Rightarrow 0_B \in B \\
\quad \wedge 1_B \in B \\
\quad \wedge (\forall\ u\ v\ w \\
\quad \bullet\ u \in B \wedge v \in B \wedge w \in B \\
\qquad \Rightarrow u +_B v \in B \\
\qquad\quad \wedge u \cdot_B v \in B \\
\qquad\quad \wedge \sim_B u \in B \\
\qquad\quad \wedge u +_B v = v +_B u \\
\qquad\quad \wedge u +_B v +_B w = (u +_B v) +_B w \\
\qquad\quad \wedge u \cdot_B v = v \cdot_B u \\
\qquad\quad \wedge u \cdot_B v \cdot_B w = (u \cdot_B v) \cdot_B w \\
\qquad\quad \wedge u \cdot_B (v +_B w) = u \cdot_B v +_B u \cdot_B w \\
\qquad\quad \wedge u \cdot_B (v +_B w) = u \cdot_B v +_B u \cdot_B w \\
\qquad\quad \wedge u +_B v \cdot_B w = (u +_B v) \cdot_B (u +_B w) \\
\qquad\quad \wedge u \cdot_B (u +_B v) = u \\
\qquad\quad \wedge u +_B u \cdot_B v = u \\
\qquad\quad \wedge u +_B \sim_B u = 1_B \\
\qquad\quad \wedge u \cdot_B \sim_B u = 0_B)
\end{array}
$$

BA_lem4 $\vdash \forall$ *BA B* \$+$_B$ \$.$_B$ $\sim_B$ 0_B 1_B

$$
\begin{array}{l}
\bullet\ \textit{BooleanAlgebra BA} \\
\qquad \wedge\ \textit{BA} = \textit{MkBA B} \$+_B \$.{}_B \sim_B 0_B\ 1_B \\
\Rightarrow (\forall\ u\ v\ w \\
\bullet\ u \in B \wedge v \in B \wedge w \in B \\
\qquad \Rightarrow u +_B 0_B = u \\
\qquad\quad \wedge u \cdot_B 1_B = u \\
\qquad\quad \wedge u \cdot_B u = u \\
\qquad\quad \wedge u +_B u = u \\
\qquad\quad \wedge u \cdot_B 0_B = 0_B \\
\qquad\quad \wedge u +_B 1_B = 1_B)
\end{array}
$$

9 The Theory bvi

9.1 Parents

ba

9.2 Constants

$\boldsymbol{Mem_{BI}}$	$('a, 'b)\ BI \to 'a \to 'a \to 'b$
$\boldsymbol{Eq_{BI}}$	$('a, 'b)\ BI \to 'a \to 'a \to 'b$
$\boldsymbol{BA_{BI}}$	$('a, 'b)\ BI \to 'b\ BA$
$\boldsymbol{U_{BI}}$	$('a, 'b)\ BI \to 'a\ \mathbb{P}$
$\boldsymbol{MkBI}$	$'a\ \mathbb{P}$ $\to 'b\ BA$ $\to ('a \to 'a \to 'b)$ $\to ('a \to 'a \to 'b)$ $\to ('a, 'b)\ BI$
$\boldsymbol{BoolInterp}$	$('a, 'b)\ BI \to BOOL$

9.3 Types

$('1, '2)\ \boldsymbol{BI}$

9.4 Fixity

Right Infix 300:

$=_{\boldsymbol{B}} \in_{\boldsymbol{B}}$

9.5 Definitions

$\boldsymbol{BI}$ $\vdash \exists\, f \bullet\ TypeDefn\ (\lambda\ x \bullet\ T)\ f$

$\boldsymbol{MkBI}$

$\boldsymbol{U_{BI}}$

$\boldsymbol{BA_{BI}}$

$\boldsymbol{Eq_{BI}}$

$\boldsymbol{Mem_{BI}}$ $\vdash \forall\ t\ x1\ x2\ x3\ x4$
$\bullet\ U_{BI}\ (MkBI\ x1\ x2\ x3\ x4) = x1$
$\wedge\ BA_{BI}\ (MkBI\ x1\ x2\ x3\ x4) = x2$
$\wedge\ Eq_{BI}\ (MkBI\ x1\ x2\ x3\ x4) = x3$
$\wedge\ Mem_{BI}\ (MkBI\ x1\ x2\ x3\ x4) = x4$
$\wedge\ MkBI\ (U_{BI}\ t)\ (BA_{BI}\ t)\ (Eq_{BI}\ t)\ (Mem_{BI}\ t)$
$= t$

$\boldsymbol{BoolInterp}$ $\vdash \forall\ i$
$\bullet\ BoolInterp\ i$
$\Leftrightarrow (\forall\ A\ x\ y\ z\ v\ w\ \$=_B\ \$\in_B$
$(B, \$+_B, \$._B, \sim_B, 0_B, 1_B)$
$\bullet\ A = U_{BI}\ i$
$\wedge\ MkBA\ B\ \$+_B\ \$._B\ \sim_B\ 0_B\ 1_B = BA_{BI}\ i$
$\wedge\ \$=_B = Eq_{BI}\ i$
$\wedge\ \$\in_B = Mem_{BI}\ i$

$$
\begin{array}{l}
\quad \wedge\ x \in A \\
\quad \wedge\ y \in A \\
\quad \wedge\ z \in A \\
\quad \wedge\ v \in A \\
\quad \wedge\ w \in A \\
\Rightarrow (\mathit{let}\ x \leq_B y \Leftrightarrow x \cdot_B \sim_B y = 0_B \\
\quad \mathit{in}\ x =_B x = 1_B \\
\qquad \wedge\ x =_B y = y =_B x \\
\qquad \wedge\ (x =_B y) \cdot_B (y =_B z) \leq_B x =_B z \\
\qquad \wedge\ (x \in_B y) \cdot_B (v =_B x) \cdot_B (w =_B y) \\
\qquad\quad \leq_B v \\
\qquad\quad \in_B w))
\end{array}
$$

10 The Theory memfunct

10.1 Parents

membership

11 INDEX