Pure Type Systems and HOL-Omega in ProofPower

Roger Bishop Jones

Abstract

The abstract syntax and semantics of Pure Type Systems and HOL-Omega in a formal higher-order theory of well-founded sets.

Created: 2009/8/31

Last Change Date: 2009/10/05 09:18:19

http://www.rbjones.com/rbjpub/pp/doc/t031.pdf

Id: t031.doc,v 1.3 2009/10/05 09:18:19 rbj Exp

Contents

References

[1] Peter V. Homeier. The HOL-Omega Logic. pages 244–259. 2009. also available at: `http://trustworthytools.com`.

[2] Roger Bishop Jones. Higher Order Set Theory. *RBJones.com*, 2010. `http://www.rbjones.com/rbjpub/pp/doc/t023.pdf`.

1 Introduction

This document was started after discussions with Peter Homeier at a workshop on Interactive Theorem Proving in 2009. It is intended for explorations into the formalisation of the semantics of Pure Type Systems and of Hol-Omega [1] using a well-founded theory of well-founded sets [2].

The connections betweem the Pure Type Systems and HOL-Omega are very tenuous indeed. For me they are both examples of systems which in some way or to some extent seek to mitigate the problems which arise in trying to work formally with a well-founded ontology.

This can be done either by dropping the constraint to a well-founded ontology, or by adopting various pragmatic paliatives. The Pure Type Systems include systems which have no well-founded semantics, notably Coquand's Calculus of Constructions. For this reason they are worth study to increase one's armoury of methods for constructing consistent logical systems without the draconian constraint to well-foundedness. HOL-Omega on the other hand, represents one of the many different ways of mitigating the problems arising from well-foundedness without actually disposing of it. HOL is itself such a system, its polymorphism probably representing the least one can do in the object language. Though HOL's polymorphism is of the very simplest kind, it is very effective. It is hard to improve upon and attempts to improve on it usually yield much more complex logical systems whose merits in the respects we consider are not beyond debate. HOL-Omega is such a system.

It may be worth mentioning what are the principle problems which this kind of development is intended to address. For me, they relate to formalisation "in tha large", i.e. in support for abstraction, modularity and reuse of specifications and proofs. The formalisation of abstract theories, notably of category theory, is also, whether or not thought of in the context of modularity, a test for this kind of foundational development.

The attempt to deal with these problems by foundational innovation is itself controversial. This is primarily I think because of the difficulties involved, the risk of ending up with a system which provides poor support for mainstream mathematics (such as analysis) done in the established manner, and because these matters can be addressed without resort to foundational innovations (and are usually addressed in such ways).

My own interest has primarily been in the foundational route, no doubt because I just like thinking about this problem. But I have also an interest in approaches, at least to the 'specification in the large' side of things, which are foundationally neutral, and allow interworking between logical systems.

So far as this document is concerned, it has so far gone nowhere, for I quickly realised (though not until after I had started the document) that the higher order theory of well-founded sets which I had intended to use for the purpose [2] fell short of the demands I would put upon it and would have to be upgraded. So that is where movement is taking place at present.

2 Pure Type Systems

3 HOL-Omega

SML

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

4 Syntax

The formalisation of the syntax has no particular interest. It is essentially the definition of a recursive datatype. If our proof tool supported datatype definitions then I would have used them, since it does not the easiest way to get the required dayatype is by using set theory. Note however, that I do not necessarily need to make a new type, and will only prove those theorems which I actually need.

The manual construction of the three mutually recursive datatypes in our higher order set theory is done by defining the constructors and taking the intersection of all sets closed under the constructors.

4.1 Constructors and Discriminators

4.1.1 Kinds

HOL Constant

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

$$\forall t \bullet MkKindTy\ t = (Nat_g\ 0) \mapsto_g t$$

HOL Constant

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

$$\forall k \bullet IsKindTy\ k \Leftrightarrow \exists t \bullet k = MkKindTy\ t$$

HOL Constant

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

$$\forall n \bullet MkKindVar\ n = (Nat_g\ 1) \mapsto_g n$$

HOL Constant

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

$$\forall k \bullet IsKindVar\ k \Leftrightarrow \exists n \bullet k = MkKindVar\ n$$

HOL Constant

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

$$\forall lr \bullet\ MkKindArr\ lr = (Nat_g\ 2) \mapsto_g (MkPair_g\ lr)$$

HOL Constant

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

$$\forall k \bullet\ IsKindArr\ k \Leftrightarrow \exists t \bullet\ k = MkKindArr\ t$$

HOL Constant

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

$$\forall k \bullet\ IsKind\ k \Leftrightarrow IsKindTy\ k \vee IsKindVar\ k \vee IsKindArr\ k$$

4.1.2 Types

HOL Constant

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

$$\forall nkr \bullet\ MkTypeVar\ nkr = (Nat_g\ 3) \mapsto_g (MkTriple_g\ nkr)$$

HOL Constant

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

$$\forall t \bullet\ IsTypeVar\ t \Leftrightarrow \exists p \bullet\ t = MkTypeVar\ p$$

HOL Constant

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

$$\forall nkr \bullet\ MkTypeCon\ nkr = (Nat_g\ 4) \mapsto_g (MkTriple_g\ nkr)$$

HOL Constant

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

$$\forall t \bullet\ IsTypeCon\ t \Leftrightarrow \exists p \bullet\ t = MkTypeCon\ p$$

HOL Constant

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

$$\forall fa \bullet\ MkTypeApp\ fa = (Nat_g\ 5) \mapsto_g (MkPair_g\ fa)$$

HOL Constant

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

$$\forall a \bullet\ IsTypeApp\ a \Leftrightarrow \exists p \bullet\ a = MkTypeApp\ p$$

HOL Constant

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

$$\forall vb \bullet\ MkTypeAbs\ vb = (Nat_g\ 6) \mapsto_g (MkPair_g\ vb)$$

HOL Constant

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

$$\forall a \bullet\ IsTypeAbs\ a \Leftrightarrow \exists p \bullet\ a = MkTypeAbs\ p$$

HOL Constant

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

$$\forall vb \bullet\ MkTypeUniv\ vb = (Nat_g\ 7) \mapsto_g (MkPair_g\ vb)$$

HOL Constant

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

$$\forall t \bullet\ IsTypeUniv\ t \Leftrightarrow \exists p \bullet\ t = MkTypeUniv\ p$$

HOL Constant

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

$$\forall t \bullet\ IsType\ t \Leftrightarrow IsTypeVar\ t \lor IsTypeCon\ t \lor IsTypeApp\ t \lor IsTypeAbs\ t \lor IsTypeUniv\ t$$

4.1.3 Terms

HOL Constant

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

$$\forall nt \bullet\ MkTermVar\ nt = (Nat_g\ 8) \mapsto_g (MkPair_g\ nt)$$

HOL Constant

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

$$\forall t \bullet\ IsTermVar\ t \Leftrightarrow \exists p \bullet\ t = MkTermVar\ p$$

HOL Constant

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

$$\forall nt \bullet\ MkTermCon\ nt = (Nat_g\ 9) \mapsto_g (MkPair_g\ nt)$$

HOL Constant

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

$$\forall t \bullet\ IsTermCon\ t \Leftrightarrow \exists p \bullet\ t = MkTermCon\ p$$

HOL Constant

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

$$\forall fa \bullet\ MkTermApp\ fa = (Nat_g\ 10) \mapsto_g (MkPair_g\ fa)$$

HOL Constant

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

$$\forall a \bullet\ IsTermApp\ a \Leftrightarrow \exists p \bullet\ a = MkTermApp\ p$$

HOL Constant

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

$$\forall vb \bullet\ MkTermAbs\ vb = (Nat_g\ 11) \mapsto_g (MkPair_g\ vb)$$

HOL Constant

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

$$\forall a \bullet\ IsTermAbs\ a \Leftrightarrow \exists p \bullet\ a = MkTermAbs\ p$$

HOL Constant

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

$$\forall fa \bullet\ MkTermAppType\ fa = (Nat_g\ 12) \mapsto_g (MkPair_g\ fa)$$

HOL Constant

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

$$\forall a \bullet\ IsTermAppType\ a \Leftrightarrow \exists p \bullet\ a = MkTermAppType\ p$$

HOL Constant

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

$$\forall vb \bullet\ MkTermAbsType\ vb = (Nat_g\ 13) \mapsto_g (MkPair_g\ vb)$$

HOL Constant

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

$$\forall a \bullet\ IsTermAbsType\ a \Leftrightarrow \exists p \bullet\ a = MkTermAbsType\ p$$

HOL Constant

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

$$\forall t \bullet\ IsTerm\ t \Leftrightarrow IsTermVar\ t \vee IsTermCon\ t \vee IsTermApp\ t \vee IsTermAbs\ t \vee IsTermAppType\ t \vee$$

4.2 The Inductive Definition of Syntax

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

The closure condition is:

HOL Constant

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

$$\begin{array}{ll}
 & \forall\ s \bullet\ HolwSynClosed\ s \Leftrightarrow \\
 & \quad (\forall t \bullet\ t \in s \wedge IsType\ t \Rightarrow \\
 & \qquad MkKindTy\ t \in s) \\
\wedge & \quad (\forall p \bullet\ MkKindVar\ p \in s) \\
\wedge & \quad (\forall k1\ k2 \bullet\ k1 \in s \wedge k2 \in s \wedge IsKind\ k1 \wedge IsKind\ k2 \Rightarrow \\
 & \qquad MkKindArr\ (k1,\ k2) \in s) \\
\wedge & \quad (\forall n\ k\ r \bullet\ k \in s \wedge IsKind\ k \Rightarrow \\
 & \qquad MkTypeVar\ (Nat_g\ n,\ k,\ Nat_g\ r) \in s) \\
\wedge & \quad (\forall n\ k\ r \bullet\ k \in s \wedge IsKind\ k \Rightarrow \\
 & \qquad MkTypeCon\ (Nat_g\ n,\ k,\ Nat_g\ r) \in s) \\
\wedge & \quad (\forall f\ a \bullet\ f \in s \wedge a \in s \wedge IsType\ f \wedge IsType\ a \Rightarrow \\
 & \qquad MkTypeApp\ (f,\ a) \in s) \\
\wedge & \quad (\forall v\ b \bullet\ v \in s \wedge b \in s \wedge IsTypeVar\ v \wedge IsType\ b \Rightarrow \\
 & \qquad MkTypeAbs\ (v,\ b) \in s) \\
\wedge & \quad (\forall v\ b \bullet\ v \in s \wedge b \in s \wedge IsTypeVar\ v \wedge IsType\ b \Rightarrow \\
 & \qquad MkTypeUniv\ (v,\ b) \in s) \\
\wedge & \quad (\forall n\ t \bullet\ t \in s \wedge IsType\ t \Rightarrow \\
 & \qquad MkTermVar\ (Nat_g\ n,\ t) \in s) \\
\wedge & \quad (\forall n\ t \bullet\ t \in s \wedge IsType\ t \Rightarrow \\
 & \qquad MkTermCon\ (Nat_g\ n,\ t) \in s) \\
\wedge & \quad (\forall f\ a \bullet\ f \in s \wedge a \in s \wedge IsTerm\ f \wedge IsTerm\ a \Rightarrow \\
 & \qquad MkTermApp\ (f,\ a) \in s) \\
\wedge & \quad (\forall v\ b \bullet\ v \in s \wedge b \in s \wedge IsTermVar\ v \wedge IsTerm\ b \Rightarrow \\
 & \qquad MkTermAbs\ (v,\ b) \in s) \\
\wedge & \quad (\forall f\ a \bullet\ f \in s \wedge a \in s \wedge IsTerm\ f \wedge IsType\ a \Rightarrow \\
 & \qquad MkTermAppType\ (f,\ a) \in s) \\
\wedge & \quad (\forall v\ b \bullet\ v \in s \wedge b \in s \wedge IsTypeVar\ v \wedge IsTerm\ b \Rightarrow \\
 & \qquad MkTermAbsType\ (v,\ b) \in s)
\end{array}$$

The well-formed syntax is then the smallest set closed under these constructions (which might be a "class").

HOL Constant

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

$$HolwSyntax = \bigcap \{x \mid HolwSynClosed\ x\}$$

5 Proof Contexts

SML

```
(* add_pc_thms "'holw" []; *)

(* add_pc_thms "'holw" []; *)
commit_pc "'holw";

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

force_new_pc "holw1";
merge_pcs ["misc21", "'holw"] "holw1";
commit_pc "holw1";
```

SML

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

6 The Theory holw

6.1 Parents

misc2

6.2 Constants

$\boldsymbol{MkKindTy}$	$GS \to GS$
$\boldsymbol{IsKindTy}$	$GS \to BOOL$
$\boldsymbol{MkKindVar}$	$GS \to GS$
$\boldsymbol{IsKindVar}$	$GS \to BOOL$
$\boldsymbol{MkKindArr}$	$GS \times GS \to GS$
$\boldsymbol{IsKindArr}$	$GS \to BOOL$
$\boldsymbol{IsKind}$	$GS \to BOOL$
$\boldsymbol{MkTypeVar}$	$GS \times GS \times GS \to GS$
$\boldsymbol{IsTypeVar}$	$GS \to BOOL$
$\boldsymbol{MkTypeCon}$	$GS \times GS \times GS \to GS$
$\boldsymbol{IsTypeCon}$	$GS \to BOOL$
$\boldsymbol{MkTypeApp}$	$GS \times GS \to GS$
$\boldsymbol{IsTypeApp}$	$GS \to BOOL$
$\boldsymbol{MkTypeAbs}$	$GS \times GS \to GS$
$\boldsymbol{IsTypeAbs}$	$GS \to BOOL$
$\boldsymbol{MkTypeUniv}$	$GS \times GS \to GS$
$\boldsymbol{IsTypeUniv}$	$GS \to BOOL$
$\boldsymbol{IsType}$	$GS \to BOOL$
$\boldsymbol{MkTermVar}$	$GS \times GS \to GS$
$\boldsymbol{IsTermVar}$	$GS \to BOOL$
$\boldsymbol{MkTermCon}$	$GS \times GS \to GS$
$\boldsymbol{IsTermCon}$	$GS \to BOOL$
$\boldsymbol{MkTermApp}$	$GS \times GS \to GS$
$\boldsymbol{IsTermApp}$	$GS \to BOOL$
$\boldsymbol{MkTermAbs}$	$GS \times GS \to GS$
$\boldsymbol{IsTermAbs}$	$GS \to BOOL$
$\boldsymbol{MkTermAppType}$	$GS \times GS \to GS$
$\boldsymbol{IsTermAppType}$	$GS \to BOOL$
$\boldsymbol{MkTermAbsType}$	$GS \times GS \to GS$
$\boldsymbol{IsTermAbsType}$	$GS \to BOOL$
$\boldsymbol{IsTerm}$	$GS \to BOOL$
$\boldsymbol{HolwSynClosed}$	$GS\ \mathbb{P} \to BOOL$
$\boldsymbol{HolwSyntax}$	$GS\ \mathbb{P}$

6.3 Definitions

MkKindTy $\vdash \forall\ t\bullet\ MkKindTy\ t = Nat_g\ 0 \mapsto_g t$

IsKindTy $\vdash \forall\ k\bullet\ IsKindTy\ k \Leftrightarrow (\exists\ t\bullet\ k = MkKindTy\ t)$

MkKindVar $\vdash \forall\ n\bullet\ MkKindVar\ n = Nat_g\ 1 \mapsto_g n$

IsKindVar $\vdash \forall\ k\bullet\ IsKindVar\ k \Leftrightarrow (\exists\ n\bullet\ k = MkKindVar\ n)$

MkKindArr $\vdash \forall\ lr\bullet\ MkKindArr\ lr = Nat_g\ 2 \mapsto_g MkPair_g\ lr$

IsKindArr $\vdash \forall\ k\bullet\ IsKindArr\ k \Leftrightarrow (\exists\ t\bullet\ k = MkKindArr\ t)$

IsKind $\vdash \forall\ k$
$\bullet\ IsKind\ k \Leftrightarrow IsKindTy\ k \vee IsKindVar\ k \vee IsKindArr\ k$

MkTypeVar $\vdash \forall\ nkr\bullet\ MkTypeVar\ nkr = Nat_g\ 3 \mapsto_g MkTriple_g\ nkr$

IsTypeVar $\vdash \forall\ t\bullet\ IsTypeVar\ t \Leftrightarrow (\exists\ p\bullet\ t = MkTypeVar\ p)$

MkTypeCon $\vdash \forall\ nkr\bullet\ MkTypeCon\ nkr = Nat_g\ 4 \mapsto_g MkTriple_g\ nkr$

IsTypeCon $\vdash \forall\ t\bullet\ IsTypeCon\ t \Leftrightarrow (\exists\ p\bullet\ t = MkTypeCon\ p)$

MkTypeApp $\vdash \forall\ fa\bullet\ MkTypeApp\ fa = Nat_g\ 5 \mapsto_g MkPair_g\ fa$

IsTypeApp $\vdash \forall\ a\bullet\ IsTypeApp\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTypeApp\ p)$

MkTypeAbs $\vdash \forall\ vb\bullet\ MkTypeAbs\ vb = Nat_g\ 6 \mapsto_g MkPair_g\ vb$

IsTypeAbs $\vdash \forall\ a\bullet\ IsTypeAbs\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTypeAbs\ p)$

MkTypeUniv $\vdash \forall\ vb\bullet\ MkTypeUniv\ vb = Nat_g\ 7 \mapsto_g MkPair_g\ vb$

IsTypeUniv $\vdash \forall\ t\bullet\ IsTypeUniv\ t \Leftrightarrow (\exists\ p\bullet\ t = MkTypeUniv\ p)$

IsType $\vdash \forall\ t$
$\bullet\ IsType\ t$
$\Leftrightarrow IsTypeVar\ t$
$\vee\ IsTypeCon\ t$
$\vee\ IsTypeApp\ t$
$\vee\ IsTypeAbs\ t$
$\vee\ IsTypeUniv\ t$

MkTermVar $\vdash \forall\ nt\bullet\ MkTermVar\ nt = Nat_g\ 8 \mapsto_g MkPair_g\ nt$

IsTermVar $\vdash \forall\ t\bullet\ IsTermVar\ t \Leftrightarrow (\exists\ p\bullet\ t = MkTermVar\ p)$

MkTermCon $\vdash \forall\ nt\bullet\ MkTermCon\ nt = Nat_g\ 9 \mapsto_g MkPair_g\ nt$

IsTermCon $\vdash \forall\ t\bullet\ IsTermCon\ t \Leftrightarrow (\exists\ p\bullet\ t = MkTermCon\ p)$

MkTermApp $\vdash \forall\ fa\bullet\ MkTermApp\ fa = Nat_g\ 10 \mapsto_g MkPair_g\ fa$

IsTermApp $\vdash \forall\ a\bullet\ IsTermApp\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTermApp\ p)$

MkTermAbs $\vdash \forall\ vb\bullet\ MkTermAbs\ vb = Nat_g\ 11 \mapsto_g MkPair_g\ vb$

IsTermAbs $\vdash \forall\ a\bullet\ IsTermAbs\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTermAbs\ p)$

MkTermAppType
$\vdash \forall\ fa\bullet\ MkTermAppType\ fa = Nat_g\ 12 \mapsto_g MkPair_g\ fa$

IsTermAppType
$\vdash \forall\ a\bullet\ IsTermAppType\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTermAppType\ p)$

MkTermAbsType
$\vdash \forall\ vb\bullet\ MkTermAbsType\ vb = Nat_g\ 13 \mapsto_g MkPair_g\ vb$

IsTermAbsType
$\vdash \forall\ a\bullet\ IsTermAbsType\ a \Leftrightarrow (\exists\ p\bullet\ a = MkTermAbsType\ p)$

IsTerm $\vdash \forall\ t$
$\bullet\ IsTerm\ t$
$\Leftrightarrow IsTermVar\ t$
$\vee\ IsTermCon\ t$
$\vee\ IsTermApp\ t$
$\vee\ IsTermAbs\ t$
$\vee\ IsTermAppType\ t$
$\vee\ IsTermAbsType\ t$

HolwSynClosed

$$
\begin{array}{l}
\vdash \forall\ s \\
\quad \bullet\ \mathit{HolwSynClosed}\ s \\
\qquad \Leftrightarrow (\forall\ t \bullet\ t \in s \wedge \mathit{IsType}\ t \Rightarrow \mathit{MkKindTy}\ t \in s) \\
\qquad\quad \wedge (\forall\ p \bullet\ \mathit{MkKindVar}\ p \in s) \\
\qquad\quad \wedge (\forall\ \mathit{k1}\ \mathit{k2} \\
\qquad\quad \bullet\ \mathit{k1} \in s \wedge \mathit{k2} \in s \wedge \mathit{IsKind}\ \mathit{k1} \wedge \mathit{IsKind}\ \mathit{k2} \\
\qquad\qquad \Rightarrow \mathit{MkKindArr}\ (\mathit{k1},\ \mathit{k2}) \in s) \\
\qquad\quad \wedge (\forall\ n\ k\ r \\
\qquad\quad \bullet\ k \in s \wedge \mathit{IsKind}\ k \\
\qquad\qquad \Rightarrow \mathit{MkTypeVar}\ (\mathit{Nat}_g\ n,\ k,\ \mathit{Nat}_g\ r) \in s) \\
\qquad\quad \wedge (\forall\ n\ k\ r \\
\qquad\quad \bullet\ k \in s \wedge \mathit{IsKind}\ k \\
\qquad\qquad \Rightarrow \mathit{MkTypeCon}\ (\mathit{Nat}_g\ n,\ k,\ \mathit{Nat}_g\ r) \in s) \\
\qquad\quad \wedge (\forall\ f\ a \\
\qquad\quad \bullet\ f \in s \wedge a \in s \wedge \mathit{IsType}\ f \wedge \mathit{IsType}\ a \\
\qquad\qquad \Rightarrow \mathit{MkTypeApp}\ (f,\ a) \in s) \\
\qquad\quad \wedge (\forall\ v\ b \\
\qquad\quad \bullet\ v \in s \wedge b \in s \wedge \mathit{IsTypeVar}\ v \wedge \mathit{IsType}\ b \\
\qquad\qquad \Rightarrow \mathit{MkTypeAbs}\ (v,\ b) \in s) \\
\qquad\quad \wedge (\forall\ v\ b \\
\qquad\quad \bullet\ v \in s \wedge b \in s \wedge \mathit{IsTypeVar}\ v \wedge \mathit{IsType}\ b \\
\qquad\qquad \Rightarrow \mathit{MkTypeUniv}\ (v,\ b) \in s) \\
\qquad\quad \wedge (\forall\ n\ t \\
\qquad\quad \bullet\ t \in s \wedge \mathit{IsType}\ t \\
\qquad\qquad \Rightarrow \mathit{MkTermVar}\ (\mathit{Nat}_g\ n,\ t) \in s) \\
\qquad\quad \wedge (\forall\ n\ t \\
\qquad\quad \bullet\ t \in s \wedge \mathit{IsType}\ t \\
\qquad\qquad \Rightarrow \mathit{MkTermCon}\ (\mathit{Nat}_g\ n,\ t) \in s) \\
\qquad\quad \wedge (\forall\ f\ a \\
\qquad\quad \bullet\ f \in s \wedge a \in s \wedge \mathit{IsTerm}\ f \wedge \mathit{IsTerm}\ a \\
\qquad\qquad \Rightarrow \mathit{MkTermApp}\ (f,\ a) \in s) \\
\qquad\quad \wedge (\forall\ v\ b \\
\qquad\quad \bullet\ v \in s \wedge b \in s \wedge \mathit{IsTermVar}\ v \wedge \mathit{IsTerm}\ b \\
\qquad\qquad \Rightarrow \mathit{MkTermAbs}\ (v,\ b) \in s) \\
\qquad\quad \wedge (\forall\ f\ a \\
\qquad\quad \bullet\ f \in s \wedge a \in s \wedge \mathit{IsTerm}\ f \wedge \mathit{IsType}\ a \\
\qquad\qquad \Rightarrow \mathit{MkTermAppType}\ (f,\ a) \in s) \\
\qquad\quad \wedge (\forall\ v\ b \\
\qquad\quad \bullet\ v \in s \wedge b \in s \wedge \mathit{IsTypeVar}\ v \wedge \mathit{IsTerm}\ b \\
\qquad\qquad \Rightarrow \mathit{MkTermAbsType}\ (v,\ b) \in s) \\
\textbf{\textit{HolwSyntax}} \quad \vdash \mathit{HolwSyntax} = \bigcap \{x | \mathit{HolwSynClosed}\ x\}
\end{array}
$$

7 INDEX