




# Homotopy Type System with Strict Equality

Namdak Tonpa  

Groupoid Infinity

Siegmentation Fault 

Groupoid Infinity

---

## Abstract

Here is presented a more modern version of `cubicaltt` called `anders`.

**2012 ACM Subject Classification** Theory of computation → Lambda calculus

**Keywords and phrases** Homotopy Type System, Cubical Type Theory

## 1 Motivation

The HTS<sup>1</sup> language proposed by Voevodsky exposes two different presheaf models of type theory: the inner one is homotopy type system presheaf that models HoTT and the outer one is traditional Martin-Löf type system presheaf that models set theory with UIP. The motivation behind this doubling is to have an ability to express semisimplicial types. Theoretical work on merging inner and outer languages was continued in 2LTT<sup>2</sup>.

While we are on our road to Lean-like tactic language, currently we are at the stage of regular cubical HTS type checker with CHM-style [5] primitives. You may try it at Github<sup>3</sup> or install through OPAM: `opam install anders`.

## 2 Syntax

The syntax resembles original syntax of the reference CCHM type checker cubicaltt, is slightly compatible with Lean syntax and contains the full set of Cubical Agda<sup>4</sup> primitives.

Here is given the mathematical pseudo-code notation of the language expressions that come immediately after parsing. The core syntax definition of HTS language E corresponds to `exp` type defined in `expr.ml` OCaml module:

$$\begin{aligned} E &:= \text{cosmos} \mid \text{var} \mid \text{MLTT} \mid \text{CCHM} \mid \text{HIT} \\ \text{HIT} &:= \text{inductive } E \ E \mid \text{ctor name } E \mid \text{match } E \ E \\ \text{CCHM} &:= \text{path} \mid I \mid \text{part} \mid \text{sub} \mid \text{kan} \mid \text{glue} \\ \text{MLTT} &:= \pi \mid \text{sigma} \mid \text{id} \end{aligned}$$

Further Menhir BNF notation will be used to describe the top-level language parser as type checker is written in OCaml.

---

<sup>1</sup> <https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/HTS.pdf> – HTS

<sup>2</sup> <https://arxiv.org/pdf/1705.03307.pdf> – 2LTT

<sup>3</sup> <https://github.com/groupoid/anders>

<sup>4</sup> <https://staff.math.su.se/anders.mortberg/papers/cubicalagda.pdf>

```

cosmos := Uj | Vk
var := var name | hole
pi :=  $\Pi$  name E E |  $\lambda$  name E E | E E
sigma :=  $\Sigma$  name E E | (E, E) | E.1 | E.2
id := Id E | ref E | idJ E
path := Path E | Ei | E @ E
I := I | 0 | 1 | E  $\vee$  E | E  $\wedge$  E |  $\neg$ E
part := Partial E E | [ (E = I)  $\rightarrow$  E, ... ]
sub := inc E | ouc E | E [ I |  $\neg$   $\rightarrow$  E ]
kan := transp E E | hcomp E
glue := Glue E | glue E | unglue E E

```

**Keywords.** The words of a top-level language (file or repl) consist of keywords or identifiers. The keywords are following: **module**, **where**, **import**, **option**, **def**, **axiom**, **postulate**, **theorem**, (, ), [, ], <, >, /, **.1**, **.2**,  $\Pi$ ,  $\Sigma$ , ,,  $\lambda$ ,  $\times$ ,  $\rightarrow$ ,  $\vdash$ ,  $\equiv$ ,  $\vdash$ , **U**, **V**,  $\vee$ ,  $\wedge$ ,  $\neg$ , **+**, **@**, **PathP**, **transp**, **hcomp**, **zero**, **one**, **Partial**, **inc**, **ouc**, **interval**, **inductive**, **Glue**, **glue**, **unglue**.

**Identifiers.** Identifiers support UTF-8. Identifiers couldn't start with :, -,  $\rightarrow$ . Sample identifiers:  $\neg$ -of- $\vee$ ,  $1 \rightarrow 1$ , is-?, =, \$~!005x,  $\infty$ ,  $x \rightarrow \text{Nat}$

**Modules.** Modules represent files with declarations. More accurate, BNF notation of module consists of imports, options and declarations.

**Imports.** The import construction supports file folder structure (without file extensions) by using reserved symbol / for hierarchy walking.

**Options.** Each option holds bool value. Language supports following options: 1) girard (enables  $U : U$ ); 2) pre-eval (normalization cache); 3) impredicative (infinite hierarchy with impredicativity rule); In Anders you can enable or disable language core types, adjust syntaxes or tune inner variables of the type checker. Here is the example how to setup minimal core able to prove internalization of MLTT-73 variation (Path instead of Id and no inductive types, see base library):

```

option HIT false
option CCHM false
option MLTT true

```

In order to turn HIT into ordinary CiC calculus you may say:

**Declarations.** Language supports following top level declarations: 1) axiom (non-computable declaration that breakes normalization); 2) postulate (alternative or inverted axiom that can preserve consistency); 3) definition (almost any explicit term or type in type theory); 5) lemma (helper in big game). 4) theorem (something valuable or complex enough). Sample declarations. For example, signature isProp ( $A : U$ ) of type U could be defined as normalization-blocking axiom without proof-term or by providing proof-term as definition.

```

axiom isProp (A : U) : U
def isSet (A : U) : U
:=  $\Pi$  (a b : A) (x y : Path A a b), Path (Path A a b) x y

```

In this example ( $A : U$ ), ( $a\ b : A$ ) and ( $x\ y : \text{Path } A\ a\ b$ ) are called telescopes. Each telescope consists of a series of lenses or empty. Each lense provides a set of variables of the

same type. Telescope defines parameters of a declaration. Types in a telescope, type of a declaration and a proof-terms are a language expressions `exp1`.

Expressions. All atomic language expressions are grouped by four categories: `exp0` (pair constructions), `exp1` (non neutral constructions), `exp2` (path and pi applications), `exp3` (neutral constructions).

The LR parsers demand to define `exp1` as expressions that cannot be used (without a parens enclosure) as a right part of left-associative application for both Path and Pi lambdas.

Universe indecies  $U_j$  (inner fibrant),  $V_k$  (outer pretypes) and  $S$  (outer strict omega) are using unicode subscript letters that are already processed in lexer.

### 3 Semantics

The idea is to have a unified layered type checker, so you can disbale/enable any MLTT-style inference, assign types to universes and enable/disable hierachies. This will be done by providing linking API for pluggable presheaf modules. We selected 5 levels of type checker awareness from universes and pure type systems up to synthetic language of homotopy type theory. Each layer corresponds to its presheaves with separate configuration for universe hierarchies.

```

inductive lang : U :=
  | UNI: cosmos → lang
  | PI: pure lang → lang
  | SIGMA: total lang → lang
  | ID: uip lang → lang
  | PATH: homotopy lang → lang
  | GLUE: gluening lang → lang
  | HIT: hit lang → lang
    
```

We want to mention here with homage to its authors all categorical models of dependent type theory: Comprehension Categories (Grothendieck, Jacobs), LCCC (Seely), D-Categories and CwA (Cartmell), CwF (Dybjer), C-Systems (Voevodsky), Natural Models (Awodey). While we can build some transports between them, we leave this excercise for our mathematical components library.

We will use here the Coquand's notation for Presheaf Type Theories in terms of restriction maps.

#### 3.1 Universe Hierarchies

Language supports Agda-style hierarchy of universes: fibrant ( $U$ ), interval pretypes ( $V$ ) and strict omega with explicit level manipulation. All universes are bounded with preorder

$$Fibrant_j \prec Pretypes_k \prec Strict_l, \quad (1)$$

in which  $j, k, l$  are bounded with equation:

$$j < k < l. \quad (2)$$

Large elimination to upper universes is prohibited. This is extendable to Agda model:

```

inductive cosmos : U :=
  | prop: nat → cosmos
  | fibrant: nat → cosmos
  | pretypes: nat → cosmos
  | strict: nat → cosmos
  | omega: cosmos
  | lock: cosmos

```

## 3.2 Dependent Types

## 3.3 Path Equality