



MLTT — CT — Proof Theory

Syntax — Semantics — Logic

$x : A$ — x is a object of type A
 $y = [x : A]$ — x and y are definitionally
equal objects of type A

$\text{Nat} : \mathcal{U} \text{ n}$ — constant functor
 $\text{List} (A : \mathcal{U} \text{ n}) : \mathcal{U} \text{ n}$ — functor
 $\text{List Nat} : \mathcal{U} \text{ n}$ — constant functor

$\text{List } A = 1 \longrightarrow \text{List } A \longrightarrow \text{List } A \longleftarrow A : \mathcal{U}$
 $\text{nil} : 1 \longrightarrow \text{List } A$
 $\text{cons} : A \longrightarrow \text{List } A \longrightarrow \text{List } A$

$\text{Nat} = 1 \longrightarrow \mathbb{N} \longrightarrow \mathbb{N} : \mathcal{U}$
 $\text{zero} : 1 \longrightarrow \mathbb{N}$
 $\text{succ} : \mathbb{N} \longrightarrow \mathbb{N}$

Pure Type System

Infinity Topoi

Grothendieck Universum

$$S (n : \text{nat}) = U \ n$$

$$A_1 (n \ m : \text{nat}) = U \ n : U \ m \text{ where } m > n \text{ — cumulative}$$

$$R_1 (m \ n : \text{nat}) = U \ m \longrightarrow U \ n : U \ (\max m \ n) \text{ — predicative}$$

$$U_0 : U_1 : U_2 : U_3 : \dots \infty$$

U_0 — propositions

U_1 — sets

U_2 — types

U_3 — sorts

$$A_2 (n : \text{nat}) = U \ n : U \ (n + 1) \text{ — non-cumulative}$$

$$R_2 (m \ n : \text{nat}) = U \ m \longrightarrow U \ n : U \ n \text{ — impredicative}$$

$$\text{Prop} = \text{Large } \Omega_0 = U_0$$

$$\Sigma = \text{Large } \Omega_2 = U_2$$

Pi Type

Functional Completeness

data $O_1 := U : \text{nat} \rightarrow O_1$
| Var: Ident $\rightarrow O_1$
| App: $O_1 \rightarrow O_1 \rightarrow O_1$
| Lambda: Binder $\rightarrow O_1 \rightarrow O_1 \rightarrow O_1$
| Arrow: $O_1 \rightarrow O_1 \rightarrow O_1$
| Pi: name $\rightarrow O_1 \rightarrow O_1 \rightarrow O_1$.

$\forall x: A, B x : U$ — formation rule
 $\lambda x: A, b : B x$ — introduction
app f a : B x — elimination
app ($\lambda o:A, b$) a = b [a/o] : B x
— computation

record Pi (A: Type) :=
intro: (A \rightarrow Type) \rightarrow Type :=
fun: (B: A \rightarrow Type) $\rightarrow \forall (a: A) \rightarrow B a \rightarrow$ intro B
app: (B: A \rightarrow Type) \rightarrow intro B $\rightarrow \forall (a: A) \rightarrow B a$
app-fun (B: A \rightarrow Type) (f: $\forall (a: A) \rightarrow B a$): $\forall (a: A) \rightarrow$ app (fun f) a ==> f a
fun-app (B: A \rightarrow Type) (p: intro B): fun ($\lambda (a: A) \rightarrow$ app p a) ==> p

Sigma Types

Contextual Completeness

`data` $O_2 := O_1$
| Sigma: $\text{name} \rightarrow O_2 \rightarrow O_2 \rightarrow O_2$
| Pair: $O_2 \rightarrow O_2 \rightarrow O_2$
| Fst: $O_2 \rightarrow O_2$
| Snd: $O_2 \rightarrow O_2$.

$\Sigma x: A, B x : U$ — formation rule
pair $(x : A) (y : B x)$ — introduction
pr1 $s : A$ — elimination
pr2 $s : B x$ — elimination

Identity Type a la Martin-Löf

record Id (A: Type): Type :=

Id: A → A → Type

refl (a: A): Id a a

Predicate: $\forall (x,y: A) \rightarrow \text{Id } x \ y \rightarrow \text{Type}$

Forall (C: Predicate): $\forall (x,y: A) \rightarrow \forall (p: \text{Id } x \ y) \rightarrow C \ x \ y \ p$

Δ (C: Predicate): $\forall (x: A) \rightarrow C \ x \ x \ (\text{refl } x)$

axiom-J (C: Predicate): $\Delta \ C \rightarrow \text{Forall } C$

computation (C: Predicate) (t: $\Delta \ C$): $\forall (x: A) \rightarrow (J \ C \ t \ x \ x \ (\text{refl } x)) ==> (t \ x) \)$

record Subst (A: Type): Type :=

intro (P (a: A): Type) (a1,a2: A) : Id a1 a2 → P a1 → P a2 :=

Id.axiom-J ($\lambda a1 \ a2 \ p12 \rightarrow P \ a1 \rightarrow P \ a2$))

K UIP Congruence

record UIP (A: Type): Type :=

intro (A: Type) (a,b: A) (x,y: Id a b) : Id (Id A a b) x y)

record K (A: Type): Type :=

PredicateK: $\forall (a: A) \rightarrow \text{Id } a \ a \rightarrow \text{Type}$

ForallK (C: PredicateK): $\forall (a: A) \rightarrow \forall (p: \text{Id } a \ a) \rightarrow C \ a \ p$

ΔK (C: PredicateK) : $\forall (a: A) \rightarrow C \ a \ (\text{Id.refl } a)$

axiom-K (C: Predicate): $\Delta K \ C \rightarrow \text{ForallK } C$

define Respect (A,B: Type) (C: A \rightarrow Type) (D: B \rightarrow Type) (R: A \rightarrow B \rightarrow Prop)

(Ro: $\forall (x: A) (y: B) \rightarrow C \ x \rightarrow D \ y \rightarrow \text{Prop}$) : ($\forall (x: A) \rightarrow C \ x$) \rightarrow ($\forall (x: B) \rightarrow D \ x$) \rightarrow Prop

:= $\lambda (f,g: \text{Type} \rightarrow \text{Type}) \rightarrow (\forall (x,y: \text{Type}) \rightarrow R \ x \ y) \rightarrow \text{Ro } x \ y \ (f \ x) \ (g \ y)$

Category

record Cat: U :=

Ob: U

Hom: (dom,cod: Ob) → Setoid

Id: (x: Ob) → Hom x x

Comp: (x,y,z: Ob) → Hom x y → Hom y z → Hom x z

Dom₁_o: (x,y: Ob) (f: Hom x y) → (Hom.Equ x y (Comp x x y id f) f)

Cod₁_o: (x,y: Ob) (f: Hom x y) → (Hom.Equ x y (Comp x y y f id) f)

Subst₁: (x,y,z: Ob) → Proper (Respect Equ (Respect Equ Equ)) (Comp x y z)

Subst_o: (x,y,z: Ob) (f₁,f₂: Hom x y) (g₁,g₂: Hom y z)

→ (Hom.Equ x y f₁ f₂) → (Hom.Equ y z g₁ g₂)

→ (Hom.Equ x z (Comp x y z f₁ g₁) (Comp x y z f₂ g₂))

Assoc_o: (x,y,z,w: Ob) (f: Hom x y) (g: Hom y z) (h: Hom z w)

→ (Hom.Equ x w (Comp x y w f (Comp y z w g h))

(Comp x z w (Comp x y z f g) h))

Setoid

```
record Setoid: Type :=  
  Carrier: Type  
  Equ: Carrier → Carrier → Prop  
  Refl: (x: Carrier) → Equ x x  
  Trans: (x1,x2,x3: Carrier) → Equ x1 x2 → Equ x2 x3 → Equ x1 x3  
  Sym: (x1,x2: Carrier) → Equ x1 x2 → Equ x2 x1
```

Application 1. Logic

`data` Proper (A: Type) (R: A → A → Prop) (m: A): Prop := intro: R m m

`data` Inhabited (A: Type): Prop := intro: A → Inhabited A

`data` True: Prop := intro: () → True

`data` False: Prop := ()

`data` Eq (A : Type): A → A → Type := refl: $\forall (x: A) \rightarrow \text{Eq } A \ x \ x$

`data` Exists (A: Type): A → Type → Type := intro: $\forall (P: A \rightarrow \text{Type}) \rightarrow$
 $\forall (x: A) \rightarrow P \ x \rightarrow \text{Exists } A \ P$

`record` SKI := S : (p → q → r) → (p → q) → p → r
 K : p → (q → p)
 I : p → p

Application 2. Control

record pure (P: Type → Type) (A: Type): Type := return: P A

record functor (F: Type → Type) (A,B: Type): Type := map: (A → B) → F A → F B

record applicative (F: Type → Type) (A,B: Type): Type :=

pure: pure F A

functor: functor F A B

ap: F (A → B) → F A → F B

record monad (F: Type → Type) (A,B: Type): Type :=

pure: pure F A

functor: functor F A B

join: F (F A) → F B

```
import Strings.String.
```

```
data Loc: Type := intro: string → nat → nat → Loc.
```

```
define Ident      := string.
```

```
define Label      := string.
```

```
define Binder     := prod Ident Loc.
```

```
define Branch     := prod Label (prod (list Binder) Type).
```

```
define NamedRec   := list (prod Binder Type).
```

```
define NamedSum   := list (prod Binder NamedRec).
```

```
define DecList    := list (prod Binder (prod Type Type)).
```

```
data Decls        := Intro: DecList → Decls
```

```
    | Opaque: Binder → Decls
```

```
    | Transparent: Binder → Decls.
```

Core

Infinity Language

data Infinity := O₂

- | Where: MLTT → Decls → MLTT
- | Con: Label → list MLTT → MLTT
- | Split: Loc → list Branch → MLTT
- | Sum: Binder → NamedSum → MLTT
- | HIT: HomotopyCalculus → MLTT
- | PI: PiCalculus → MLTT
- | EFF: EffectCalculus → MLTT
- | STREAM: StreamCalculus → MLTT.

Homotopy Calculus

import core.

data HomotopyCalculus :=

Id	Refl	Inh	Inc
Squash	InhRec	TransU	TransInvU
TransURef	Singl	MapOnPath	AppOnPath
HExt	EquivEq	EquivEqRef	TransUEquivEq
IdP	MapOnPathD	IdS	MapOnPathS
AppOnPathD	Circle	Base	HLoop
CircleRec	I	IO	I1
Line	IntRec	Undef: Loc → HomotopyCalculus.	

$\text{Id} : (A : \mathcal{U}) (a\ b : A) \rightarrow \mathcal{U}$

$\text{IdP} : (A\ B : \mathcal{U}) \rightarrow \text{Id}\ \mathcal{U}\ A\ B \rightarrow A \rightarrow B \rightarrow \mathcal{U}$

$\text{refl} : (A : \mathcal{U}) (a : A) \rightarrow \text{Id}\ A\ a\ a$

$\text{inh} : \mathcal{U} \rightarrow \mathcal{U}$

$\text{inc} : (A : \mathcal{U}) \rightarrow A \rightarrow \text{inh}\ A$

$\text{squash} : (A : \mathcal{U}) \rightarrow \text{prop}\ (\text{inh}\ A)$

$\text{inhrec} : (A : \mathcal{U}) (B : \mathcal{U}) (p : \text{prop}\ B) (f : A \rightarrow B) (a : \text{inh}\ A) \rightarrow B$

$\text{contrSingl} : (A : \mathcal{U}) (a\ b : A) (p : \text{Id}\ A\ a\ b) \rightarrow \text{Id}\ (\text{singl}\ A\ a)\ (a, \text{refl}\ A\ a)\ (b, p)$

$\text{equivEq} : (A\ B : \mathcal{U}) (f : A \rightarrow B) (s : (y : B) \rightarrow \text{fiber}\ A\ B\ f\ y)$

$(t : (y : B) \rightarrow (v : \text{fiber}\ A\ B\ f\ y) \rightarrow$

$\text{Id}\ (\text{fiber}\ A\ B\ f\ y)\ (s\ y)\ v) \rightarrow \text{Id}\ \mathcal{U}\ A\ B$

$\text{equivEqRef} : (A : \mathcal{U}) \rightarrow (s : (y : A) \rightarrow \text{pathTo}\ A\ y) \rightarrow$

$(t : (y : A) \rightarrow (v : \text{pathTo}\ A\ y) \rightarrow$

$\text{Id}\ (\text{pathTo}\ A\ y)\ (s\ y)\ v) \rightarrow$

$\text{Id}\ (\text{Id}\ \mathcal{U}\ A\ A)\ (\text{refl}\ \mathcal{U}\ A)\ (\text{equivEq}\ A\ A\ (\text{id}\ A)\ s\ t)$

Equiv Squash

Id Inh Inc

Proposition Fibration

Path Singleton

$\text{id} : (A : U) \rightarrow A \rightarrow A$

$\text{id } A \ a = a$

$\text{sld} : (A : U) (a : A) \rightarrow \text{pathTo } A \ a$

$\text{sld } A \ a = (a, \text{refl } A \ a)$

$\text{singl} : (A : U) \rightarrow A \rightarrow U$

$\text{singl } A \ a = \text{Sigma } A \ (\text{Id } A \ a)$

$\text{pathTo} : (A:U) \rightarrow A \rightarrow U$

$\text{pathTo } A = \text{fiber } A \ A \ (\text{id } A)$

$\text{IdS} : (A : U) (F : A \rightarrow U) (a_0 \ a_1 : A) (p : \text{Id } A \ a_0 \ a_1) \rightarrow F \ a_0 \rightarrow F \ a_1 \rightarrow U$

$\text{IdS } A \ F \ a_0 \ a_1 \ p = \text{IdP } (F \ a_0) \ (F \ a_1) \ (\text{mapOnPath } A \ U \ F \ a_0 \ a_1 \ p)$

$\text{prop} : U \rightarrow U$

$\text{prop } A = (a \ b : A) \rightarrow \text{Id } A \ a \ b$

$\text{Sigma} : (A : U) (B : A \rightarrow U) \rightarrow U$

$\text{Sigma } A \ B = (x : A) * B \ x$

$\text{fiber} : (A \ B : U) (f : A \rightarrow B) (y : B) \rightarrow U$

$\text{fiber } A \ B \ f \ y = \text{Sigma } A \ (\lambda x \rightarrow \text{Id } B \ (f \ x) \ y)$

Transport

$\text{transport} : (A\ B : U) \rightarrow \text{Id}\ U\ A\ B \rightarrow A \rightarrow B$

$\text{transpInv} : (A\ B : U) \rightarrow \text{Id}\ U\ A\ B \rightarrow B \rightarrow A$

$\text{transportRef} : (A : U) (a : A) \rightarrow \text{Id}\ A\ a\ (\text{transport}\ A\ A\ (\text{refl}\ U\ A)\ a)$

$\text{transpEquivEq} : (A\ B : U) \rightarrow (f : A \rightarrow B) (s : (y : B) \rightarrow \text{fiber}\ A\ B\ f\ y) \rightarrow$
 $(t : (y : B) \rightarrow (v : \text{fiber}\ A\ B\ f\ y) \rightarrow \text{Id}\ (\text{fiber}\ A\ B\ f\ y)\ (s\ y)\ v) \rightarrow$
 $(a : A) \rightarrow \text{Id}\ B\ (f\ a)\ (\text{transport}\ A\ B\ (\text{equivEq}\ A\ B\ f\ s\ t)\ a)$

App

$\text{appOnPath} : (A\ B : U) (f\ g : A \rightarrow B) (a\ b : A) (q : \text{Id}\ (A \rightarrow B)\ f\ g) (p : \text{Id}\ A\ a\ b) \rightarrow \text{Id}\ B\ (f\ a)\ (g\ b)$

$\text{appOnPathD} : (A : U) (F : A \rightarrow U) (f\ g : (x : A) \rightarrow F\ x) \rightarrow \text{Id}\ ((x : A) \rightarrow F\ x)\ f\ g \rightarrow$
 $(a_0\ a_1 : A) (p : \text{Id}\ A\ a_0\ a_1) \rightarrow \text{IdS}\ A\ F\ a_0\ a_1\ p\ (f\ a_0)\ (g\ a_1)$

Map

$\text{mapOnPath} : (A\ B : U) (f : A \rightarrow B) (a\ b : A) (p : \text{Id}\ A\ a\ b) \rightarrow \text{Id}\ B\ (f\ a)\ (f\ b)$

$\text{mapOnPathD} : (A : U) (F : A \rightarrow U) (f : (x : A) \rightarrow F\ x) (a_0\ a_1 : A) (p : \text{Id}\ A\ a_0\ a_1) \rightarrow \text{IdS}\ A\ F\ a_0\ a_1\ p\ (f\ a_0)\ (f\ a_1)$

$\text{mapOnPathS} : (A : U) (F : A \rightarrow U) (C : U) (f : (x : A) \rightarrow F\ x \rightarrow C)$
 $(a_0\ a_1 : A) (p : \text{Id}\ A\ a_0\ a_1) (b_0 : F\ a_0) (b_1 : F\ a_1)$
 $(q : \text{IdS}\ A\ F\ a_0\ a_1\ p\ b_0\ b_1) \rightarrow \text{Id}\ C\ (f\ a_0\ b_0)\ (f\ a_1\ b_1)$

Extensionality

`funHExt` : $(A : U) (B : A \rightarrow U) (f\ g : (a : A) \rightarrow B\ a) \rightarrow$
 $((x\ y : A) \rightarrow (p : \text{Id}\ A\ x\ y) \rightarrow \text{IdS}\ A\ B\ x\ y\ p\ (f\ x)\ (g\ y)) \rightarrow \text{Id}\ ((y : A) \rightarrow B\ y)\ f\ g$

`record` `I` : $U :=$

`IO` : I

`I1` : I

`line` : $\text{Id}\ I\ IO\ I1$

`rec` : $(F : I \rightarrow U) (s : F\ IO) (e : F\ I1) (l : \text{IdS}\ I\ F\ IO\ I1\ \text{line}\ s\ e) (x : I) \rightarrow F\ x$

Interval

`record` `S1` : $U :=$

`base` : S^1

`loop` : $\text{Id}\ S^1\ \text{base}\ \text{base}$

`rec` : $(F : S^1 \rightarrow U) (b : F\ \text{base}) (l : \text{IdS}\ S^1\ F\ \text{base}\ \text{base}\ \text{loop}\ b\ b) (x : S^1) \rightarrow F\ x$

Circle

Ladder to computable HITs

1. Barendregt. The Lambda Calculus with Types <http://5ht.co/pts.pdf>
2. Martin-Löf. Intuitionistic Type Theory <http://5ht.co/mltt.pdf>
3. Awodey. Category Theory <http://5ht.co/cat.pdf>
4. Hermida, Jacobs. Fibrations with indeterminates <http://5ht.co/completeness.pdf>
5. Jacobs. Categorical Logic <http://5ht.co/fibrations.pdf>
6. Streicher. The groupoid interpretation of type theory <http://5ht.co/gropuoid.pdf>
7. Voevodsky et al. Homotopy Type Theory <http://5ht.co/hott.pdf>
8. Huber, Coquand. Cubical Type Theory <http://5ht.co/cubicaltt.pdf>