Homotopy Type Theory 入門

上村 太一

2017 年 8 月 7 日

概要

Homotopy Type Theory (HoTT) は Martin-Löf 依存型理論の拡張で、ホモトピー論に特化した型理論である。代数トポロジーと計算機科学の橋渡しとなる分野であり、ホモトピー論、高次圏論、数理論理学、定理証明器、依存型プログラミングなどと関わりが深い。特に定理証明器を用いた代数トポロジーおよび数学全般の形式化に使われており、HoTT による数学の基礎付けとその定理証明器上での実装は Univalent Foundations program と呼ばれる。本チュートリアルでは、HoTT の重要な概念である univalence axiom と higher inductive type およびそれらの使い方を紹介する。

1 イントロダクション

Homotopy Type Theory (HoTT) [Uni13] は Martin-Löf 依存型理論の拡張で、ホモトピー論に特化した型理論である。代数トポロジーに全く新しい、計算機で検証可能な手法を提供する一方、数理論理学や計算機科学の立場からも興味深い対象である。

HoTT には二つの重要な概念がある。一つは Voevodsky の univalence axiom (UA) である。これは同型な型を等しいとみなしてよいことを保証する公理である。このような同一視は普段の数学ではよく行われていることであるが、それを矛盾なく形式化するのはそれほど自明ではない。もう一つは higher inductive type (HIT) である。これは帰納的型の拡張で、単位円周や球面などの空間を表す型や、自由群や商型などの要素間の同一視が必要な型を構成できる。

HoTT はホモトピー論との相性が良いが、数学全般に対しても計算機で検証可能な基礎付けを与える。HoTT による数学の基礎付けとその定理証明器上での実装は Univalent Foundations program と呼ばれる。

1.1 型と空間

HoTT の基本的なアイディアは、型を空間、要素を点と解釈することである。様々な型理論の概念が表 1 のように解釈される。特に重要なのが、等式型 $a =_A a'$ を点 a から点 a' への道の空間と解釈することである。従来の数学では二つの要素が等しいことは「命題」であり、高々一つしか要素を持たない。この性質は uniqueness of identity proofs (UIP) と呼ばれる。一方、道は「構造」であり、いくつもの異なる道があり得る。この違いは型の等しさを考えるとはっきりする。数

表 1　型理論のホモトピー論的解釈

型理論	ホモトピー論
型 A	空間 A
要素 $a : A$	点 $a \in A$
依存型 $B : A \to \mathcal{U}$	ファイブレーション $B \to A$
依存和 $\sum_{x:A} B(x)$	全空間
依存積 $\prod_{x:A} B(x)$	切片の空間
等式型 $a =_A a'$	a から a' への道の空間

学では同値な構造を同一視することがしばしばあるが、等式型を命題と考えている限りこれは正当化できない。なぜなら同値射は構造であり、いくつもの異なる同値射があり得るからだ。一方、空間的な解釈の場合、各点が空間で各道が同値射であるような空間を構成することで、型の等しさと型の同値を同一視することが正当化される。このことから、型の等しさと型の同値を同一視するための公理を追加することが考えられ、それが Voevodsky の univalence axiom である。

Martin-Löf 型理論を空間的に解釈できるといっても、元々の型理論は離散的な空間しか持っていない。高次元の構造を持つ空間を作るためには higher inductive type (HIT) を使う。例えば、円周 $\mathbb{S}^1$ は基点 $\mathsf{base} : \mathbb{S}^1$ とループ $\mathsf{loop} : \mathsf{base} = \mathsf{base}$ で生成される HIT である。Univalence axiom と HIT は片方だけあってもあまり威力はない。Univalence axiom だけでは興味深い空間は作れないし、$\mathbb{S}^1$ が非自明な空間であることの証明には univalence axiom を使う。Univalence axiom と HIT の二つがあって初めて型理論が「ホモトピーらしく」なるといえる。

1.2　アウトライン

本チュートリアルでは、univalence axiom と HIT がどのようなものかを理解することを目標とする。[Uni13] に従って、型理論は努めてインフォーマルに扱い、定理証明器による形式化には立ち入らない。2 章では、 Martin-Löf 型理論の基本的な概念を紹介する。3 章で univalence axiom と HIT について述べる。4 章では、univalence と HIT をどう使うかの例として、円周の基本群が整数の加法群と同型であることを示す。各章の終わりに参考文献を挙げる他、5 章で発展的な話題を紹介する。

また、用語についてはできるだけ日本語訳をつけているが、それらが標準的な日本語訳とは限らないことに注意されたい。

1.3　参考文献等

HoTT の標準的な教科書はオンラインで入手可能 [Uni13] である。教科書では HoTT を使ってホモトピー論、圏論、集合論、実数論をどのように扱うかがインフォーマルに書かれている。定理証明器 Coq のコードも含めたイントロダクションとして [PW14] がある。

HoTT の起源は少なくとも Martin Hofmann と Thomas Streicher による型理論のグルーポイド解釈 [HS98] まで遡れる。彼らはグルーポイドのなす圏が Martin-Löf 依存型理論のモデルになることを示し、そのモデルで UIP が成り立たないことを証明した。また、集合と全単射のなすグルーポイドが今でいう univalence axiom (彼らは universe extensionality と呼んでいた) を満たすことも示している。

2005-2006 年に、Steve Awodey と Michael Warren [AW09, War08] と Vladimir Voevodsky [Voe06, Voe09, Voe10] は独立に型理論のホモトピー論的解釈に気づいた。Awodey と Warren はモデル圏 (model category) や弱分解系 (weak factorization system) を使い、Voevodsky は単体的集合 (simplicial set) の圏を使った。Voevodsky [Voe09, KLV12] は単体的集合の圏が univalence と呼ばれる性質を満たすことに気づき、univalence axiom を提唱した。Higher inductive type のアイディアは 2011 年の Oberwolfach ミーティングでの Andrej Bauer, Peter Lumsdaine, Mike Shulman, Michael Warren による議論から産まれた [Uni13, 6 章, Notes]。

2 依存型理論

この章では、依存型理論 (dependent type theory) の各概念について述べる。HoTT の土台となる依存型理論は Martin-Löf 型理論 [ML75] である。

Martin-Löf 型理論は構成的な数学の基礎付けであり、数学的対象を構成するための諸規則からなる。あらゆる数学的対象は 型 (type) を持つ。数学的対象 a が型 A を持つことを

$$a : A$$

と書き、 a は型 A の 要素 (element) であるとも言う。数学的対象は 変数 (variable) に依存する場合もあり、a が変数 x に依存するかもしれないことを強調するときは a を $a[x]$ と書く。変数には型が同じ数学的対象を 代入 (substitute) することができ、$a[x]$ 中の x に数学的対象 b を代入して得られる数学的対象を $a[b]$ と書く。二つの数学的対象が等しいことを $a \equiv b$ で表す。数学的対象 a に名前 f を付けるときには $f :\equiv a$ と書き、 f を a と定義するなどと言う。

2.1 関数

型 A, B に対し、A から B への 関数型 (function type) を $A \to B$ と書く。いつものように、変数 $x : A$ と要素 $b[x] : B$ に対し、 λ-抽象 (λ-abstraction) $\lambda(x : A).b[x] : A \to B$ が、要素 $f : A \to B$ と $a : A$ に対し、 関数適用 (application) $f(a) : B$ が定義される。α-同値、β-同値、η-同値な要素は同一視する。

α-同値　y が $b[x]$ で使われていない変数のとき、$\lambda(x : A).b[x] \equiv \lambda(y : A).b[y]$
β-同値　$(\lambda(x : A).b[x])a \equiv b[a]$
η-同値　x が f で使われていない変数のとき、$\lambda(x : A).f(x) \equiv f$

複数回の関数適用 $f(a_1) \dots (a_n)$ や λ-抽象 $\lambda(x_1 : A_1). \dots .\lambda(x_n : A_n).b$ はまとめて $f(a_1, \dots, a_n)$

や $\lambda(x_1 : A_1, \ldots, x_n : A_n)$ と書く。また、x の型が明らかな場合、λ-抽象の型注釈は省略して単に $\lambda x.b$ と書く。

2.2 宇宙

要素が型であるような型を 宇宙 (universe) という。HoTT は可算個の宇宙の階層

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \ldots$$

を持つ。$i < j$ について、$A : \mathcal{U}_i$ ならば $A : \mathcal{U}_j$ である。考えている型がすべて同じ宇宙 $\mathcal{U}_i$ に属するときは、添字を省略して単に $\mathcal{U}$ と書く。$A : \mathcal{U}$ に対し、$A \to \mathcal{U}$ の要素を A 上の 型の族 (type family) という。

2.3 依存積

依存積 (dependent product) は関数型の一般化であり、引数によって返り値が異なるかもしれない関数を要素に持つ。$A : \mathcal{U}$ と $B : A \to \mathcal{U}$ に対し、依存積 $\prod_{x:A} B(x) : \mathcal{U}$ は関数適用と λ-抽象を持つ。

関数適用 $f : \prod_{x:A} B(x)$ と $a : A$ に対し、 $f(a) : B(a)$
λ-抽象 変数 $x : A$ と要素 $b[x] : B(x)$ に対し、$\lambda(x : A).b[x] : \prod_{x:A} B(x)$

α-同値、β-同値、η-同値は普通の関数と同様。

$A, B : \mathcal{U}$ に対し、$f : \prod_{x:A} B$ は返り値が $x : A$ に依存せず、従来の関数である。この意味で、関数型は依存積の特殊な場合とみなせる。つまり、$\prod_{x:A} B \equiv (A \to B)$ である。

例 1. **恒等射** $\mathsf{id} :\equiv (\lambda X.\lambda x.x) : \prod_{X:\mathcal{U}} X \to X$
合成 $g : B \to C$ と $f : A \to B$ に対して、$g \circ f :\equiv \lambda x.g(f(x))$ とする。このとき、$(\lambda(X, Y, Z, g, f).g \circ f) : \prod_{X,Y,Z:\mathcal{U}} (Y \to Z) \to (X \to Y) \to (X \to Z)$

2.4 帰納的型

宇宙と依存積以外のほとんどの型は 帰納的型 (inductive type) として定義される。帰納的型は

1. 構成規則 (formation rule)
2. 構成子 (constructor)
3. 除去子 (eliminator)
4. 計算規則 (computation rule)

の 4 つの規則によって記述される。構成規則 はその型がとるパラメータなどを指定する。構成子 はその型の要素を作るための定数や関数である。除去子 はその型から他の型への関数を作るもので、帰納法原理 (induction principle) とも呼ばれる。計算規則 は除去子に構成子を適用したとき

にどのように簡略されるかという規則である。

いくつかの例を挙げる。

2.4.1 自然数

自然数 (natural number) 全体の型 $\mathbb{N}$ は次で定義される。

1. 構成規則
   $\mathbb{N} : \mathcal{U}$
2. 構成子
   (a) $0 : \mathbb{N}$
   (b) $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$
3. 除去子
   $\mathsf{ind}_{\mathbb{N}} : \prod_{P:\mathbb{N}\to\mathcal{U}} P(0) \to (\prod_{x:\mathbb{N}} P(x) \to P(\mathsf{suc}(x))) \to \prod_{x:\mathbb{N}} P(x)$
4. 計算規則
   (a) $\mathsf{ind}_{\mathbb{N}}(P, z, s, 0) \equiv z$
   (b) $\mathsf{ind}_{\mathbb{N}}(P, z, s, \mathsf{suc}(x)) \equiv s(x, \mathsf{ind}_{\mathbb{N}}(P, z, s, x))$

$P : \mathbb{N} \to \mathcal{U}$ を $\mathbb{N}$ 上の述語だと思うと、除去子は自然数の数学的帰納法そのものである。

$\mathbb{N}$ からの関数は通常、パターンマッチ によって定義する。

例 2. $\mathsf{double} : \mathbb{N} \to \mathbb{N}$ を次で定義する。

1. $\mathsf{double}(0) :\equiv 0$
2. $\mathsf{double}(\mathsf{suc}(n)) :\equiv \mathsf{suc}(\mathsf{suc}(\mathsf{double}(n)))$

形式的には、$\mathsf{double} :\equiv \mathsf{ind}_{\mathbb{N}}(\lambda x.\mathbb{N}, 0, \lambda(x, n).\mathsf{suc}(\mathsf{suc}(n)))$ である。

2.4.2 依存和

依存和 (dependent sum) $\sum_{x:A} B(x)$ は次で定義される。

1. 構成規則
   $A : \mathcal{U}, B : A \to \mathcal{U}$ ならば、$\sum_{x:A} B(x) : \mathcal{U}$
2. 構成子
   $\mathsf{pair} : \prod_{x:A} B(x) \to \sum_{x:A} B(x)$
3. 除去子
   $\mathsf{ind}_{\sum_{x:A} B} : \prod_{P:(\sum_{x:A} B(x))\to\mathcal{U}} (\prod_{x:A} \prod_{y:B(x)} P(\mathsf{pair}(x, y))) \to \prod_{z:\sum_{x:A} B(x)} P(z)$
4. 計算規則
   $\mathsf{ind}_{\sum_{x:A} B}(P, c, \mathsf{pair}(x, y)) \equiv c(x, y)$

$A, B : \mathcal{U}$ に対し、$A \times B \equiv \sum_{x:A} B$ とする。

例 3.

$$\mathsf{pr}_1 : (\sum_{x:A} B(x)) \to A$$
$$\mathsf{pr}_1(\mathsf{pair}(x,y)) :\equiv x$$
$$\mathsf{pr}_2 : \prod_{z:\sum_{x:A} B(x)} B(\mathsf{pr}_1(z))$$
$$\mathsf{pr}_2(\mathsf{pair}(x,y)) :\equiv y$$

2.4.3 直和

直和 (coproduct) $A + B$ は次で定義される。

1. 構成規則
   $A, B : \mathcal{U}$ ならば $A + B : \mathcal{U}$
2. 構成子
   (a) $\mathsf{inl} : A \to A + B$
   (b) $\mathsf{inr} : B \to A + B$
3. 除去子
   $\mathsf{ind}_{A+B} : \prod_{P:A+B\to\mathcal{U}}(\prod_{x:A} P(\mathsf{inl}(x))) \to (\prod_{y:B} P(\mathsf{inr}(y))) \to \prod_{z:A+B} P(z)$
4. 計算規則
   (a) $\mathsf{ind}_{A+B}(P, l, r, \mathsf{inl}(x)) \equiv l(x)$
   (b) $\mathsf{ind}_{A+B}(P, l, r, \mathsf{inr}(y)) \equiv r(y)$

2.4.4 Unit Type

Unit type $\mathbf{1}$ は次で定義される。

1. 構成規則
   $\mathbf{1} : \mathcal{U}$
2. 構成子
   $\star : \mathbf{1}$
3. 除去子
   $\mathsf{ind}_{\mathbf{1}} : \prod_{P:\mathbf{1}\to\mathcal{U}} P(\star) \to \prod_{x:\mathbf{1}} P(x)$
4. 計算規則
   $\mathsf{ind}_{\mathbf{1}}(P, t, \star) \equiv t$

2.4.5 帰納的型の族

帰納的型だけでなく、帰納的型の族も考えることができる。例えば、長さ n の A のリストの型 $\mathbf{Vec}(A, n)$ は次で定義される帰納的型の族である。

1. 構成規則
   $A : \mathcal{U}$ ならば、$\mathbf{Vec}(A) : \mathbb{N} \to \mathcal{U}$
2. 構成子
   (a) $\mathsf{nil} : \mathbf{Vec}(A, 0)$
   (b) $\mathsf{cons} : \prod_{n:\mathbb{N}} A \to \mathbf{Vec}(A, n) \to \mathbf{Vec}(A, \mathsf{suc}(n))$
3. 除去子

$$\mathsf{ind}_{\mathbf{Vec}(A)} : \prod_{P:\prod_n \mathbf{Vec}(A,n)\to\mathcal{U}} P(0, \mathsf{nil}) \to (\prod_{n:\mathbb{N},x:A,s:\mathbf{Vec}(A,n)} P(n,s) \to P(\mathsf{suc}(n), \mathsf{cons}(n,x,s))) \to \prod_{n:\mathbb{N},s:\mathbf{Vec}(A,n)} P(n,s)$$

4. 計算規則
   (a) $\mathsf{ind}_{\mathbf{Vec}(A)}(P, z, c, 0, \mathsf{nil}) \equiv z$
   (b) $\mathsf{ind}_{\mathbf{Vec}(A)}(P, z, c, \mathsf{suc}(n), \mathsf{cons}(n, x, s)) \equiv c(n, x, s, \mathsf{ind}_{\mathbf{Vec}(A)}(P, z, c, n, s))$

2.5 等式型

等式型 (identity type) $a =_A b$ は次で定義される帰納的型の族である。

1. 構成規則
   $A : \mathcal{U}$ ならば、$(-) =_A (-) : A \to A \to \mathcal{U}$
2. 構成子
   $\mathsf{refl} : \prod_{x:A} x =_A x$
3. 除去子
   $\mathsf{ind}_{=_A} : \prod_{P:\prod_{x:A,y:A} x=_A y\to\mathcal{U}} (\prod_{x:A} P(x, x, \mathsf{refl}(x))) \to \prod_{x:A,y:A,p:x=_A y} P(x, y, p)$
4. 計算規則
   $\mathsf{ind}_{=_A}(P, r, x, x, \mathsf{refl}(x)) \equiv r(x)$

添字 $_A$ はよく省略し単に $a = b$ と書く。構成子は 反射律 (reflexivity) を表す。除去子は次のように解釈できる。

- $x = y$ であるような要素 $x, y : A$ についての述語 P がすべての $x = y$ であるような (x, y) について真であることを証明するには、反射律によって $x = x$ である (x, x) について真であることを示せば十分である。

HoTT では等式型を道の空間だとみなすことから、等式型の帰納法原理は道帰納法 (path induction) とも呼ばれる。$f : \prod_{x,y:A} \prod_{p:x=y} B(x, y, p)$ の型の関数について、x と y はしばしば暗黙の引数として扱い、$f(x, y, p)$ を単に $f(p)$ と書くことがある。

2.5.1 もう一つの定義

等式型は次のようにも定義できる。

1. 構成規則
   $A : \mathcal{U}, a : A$ ならば $a = (-) : A \to \mathcal{U}$
2. 構成子
   $\mathsf{refl}(a) : a = a$
3. 除去子
   $\mathsf{ind}_{a=} : \prod_{P:\prod_{x:A} a=x\to\mathcal{U}} P(a, \mathsf{refl}(a)) \to \prod_{x:A,p:a=x} P(x,p)$
4. 計算規則
   $\mathsf{ind}_{a=}(P, r, a, \mathsf{refl}(a)) \equiv r$

この定義での除去子は 基点付き道帰納法 (based path induction) と呼ばれる。先の道帰納法と基点付き道帰納法は同値であることが知られている [Uni13, 1.12.2 節]。以降ではどちらの場合も単に道帰納法と呼ぶことにする。

2.5.2 型は ∞-グルーポイド

命題 4. $x, y, z : A$ とする。

1. $(-)^{-1} : x = y \to y = x$ という型の関数で、$(\mathsf{refl}(x))^{-1} \equiv \mathsf{refl}(x)$ となるものを作れる。
2. $(-)\cdot(-) : x = y \to y = z \to x = z$ という型の関数で、$\mathsf{refl}(x)\cdot q \equiv q$ となるものを作れる。

Proof. 道帰納法による。 □

$x =_A y$ を等式と思うなら命題 4 は $=_A$ が同値関係であることを意味する。また、表 2 のように、型 A を空間や ∞-グルーポイド (∞-groupoid) と思うこともできる。型を ∞-グルーポイドと

表 2　型は空間または ∞-グルーポイド

	等式	空間	∞-グルーポイド
$\mathsf{refl}(x)$	反射律	自明な道	恒等射
p^{-1}	対称律	逆向きの道	逆射
$p \cdot q$	推移律	道の結合	射の合成

みなしたときの射の合成の結合律 (associativity) 等も成り立つ。

命題 5. $x, y, z, w : A, p : x = y, q : y = z, r : z = w$ とする。次の型の要素を作れる。

1. $p \cdot \mathsf{refl}(y) = p$ と $\mathsf{refl}(x) \cdot p = p$
2. $p^{-1} \cdot p = \mathsf{refl}(y)$ と $p \cdot p^{-1} = \mathsf{refl}(x)$

3. $(p^{-1})^{-1} = p$
4. $(p \cdot q) \cdot r = p \cdot (q \cdot r)$

Proof. 道帰納法。 □

型が ∞-グルーポイドであることを示すにはさらに高次元の コヒーレンス則 (coherence law) を示さなければならないが、しばらくは低次元のコヒーレンス則があれば十分であるし、それらは道帰納法によって容易に示すことができる。

2.5.3 関数は関手

命題 6. $f : A \to B, x, y : A$ とする。このとき、$\mathsf{ap}(f) : x = y \to f(x) = f(y)$ という型の関数で、$\mathsf{ap}(f, \mathsf{refl}(x)) \equiv \mathsf{refl}(f(x))$ となるものを作れる。

Proof. 道帰納法。 □

命題 6 は、等式が関数適用によって保存されることを意味する。型を空間と見たときは、関数が道を道に写す、つまり HoTT での関数は 連続 (continuous) であることを意味する。また、型を ∞-グルーポイドとみなすと、関数は 関手 (functor) であるとも解釈できる。通常の圏論の記法に合わせて、$p : x = y$ に対し、$\mathsf{ap}(f, p)$ を単に $f(p)$ と書くこともある。

命題 7. $f : A \to B, x, y, z : A, p : x = y, q : y = z$ とする。このとき、次の型の要素を作れる。

1. $f(p \cdot q) = f(p) \cdot f(q)$
2. $f(p^{-1}) = f(p)^{-1}$

Proof. 道帰納法。 □

最後に、等しい要素同士は述語によって区別できないこと (indiscernability of identicals) を示す。

命題 8. $C : A \to \mathcal{U}, x, y : A$ とする。

1. $\mathsf{transport}(C) : x = y \to C(x) \to C(y)$ という型の関数で、$\mathsf{transport}(C, \mathsf{refl}(x)) \equiv \mathsf{id}_{C(x)}$ となるものを作れる。
2. $u : C(x), p : x = y$ に対し、$\mathsf{lift}(u, p) : \mathsf{pair}(x, u) = \mathsf{pair}(y, \mathsf{transport}(C, p, u))$ なる $\sum_{x:A} C(x)$ の道を作れる。

Proof. 道帰納法。 □

このことはホモトピー論または ∞-グルーポイドの言葉では、型の族 $C : A \to \mathcal{U}$ が A 上の ファイブレーション (fibration) であることを意味する。これを説明するために、グルーポイドのファイブレーションを思い出そう。

定義 9. グルーポイド間の関手 $P : \mathbf{E} \to \mathbf{B}$ がファイブレーション (fibration) であるとは、任意の $\mathbf{E}$ の対象 X と $\mathbf{B}$ の射 $f : P(X) \to J$ に対して、$\mathbf{E}$ の射 $\bar{f} : X \to f_!(X)$ で $P(\bar{f}) = f$ となるものが存在することである。

$$\begin{array}{ccc} \mathbf{E} & & X \overset{\bar{f}}{\dashrightarrow} f_!(X) \\ {\scriptstyle P}\downarrow & & \\ \mathbf{B} & & P(X) \xrightarrow[f]{} I \end{array}$$

命題 8 は射影 $\mathsf{pr}_1 : (\sum_{x:A} C(x)) \to A$ がグルーポイドのファイブレーションのように振る舞うことを示唆する。

$$\begin{array}{ccc} \sum_{x:A} C(x) & & (x,u) \overset{\mathsf{lift}(u,p)}{\dashrightarrow} (y, \mathsf{transport}(C,p,u)) \\ {\scriptstyle \mathsf{pr}_1}\downarrow & & \\ A & & x \xrightarrow[p]{} y \end{array}$$

2.6 Propositions as Types

単純型理論が命題論理と対応するように、依存型理論には述語論理との対応がある (表 3)。この

表 3　依存型理論と述語論理

型	論理
$A \to B$	$A \supset B$
$A \times B$	$A \wedge B$
$A + B$	$A \vee B$
$a = b$	$a = b$
$\prod_{x:A} B(x)$	$\forall_{x:A} B(x)$
$\sum_{x:A} B(x)$	$\exists_{x:A} B(x)$

対応関係から、「型 A の要素を見つける」ことを「A を証明する」と言うことがある。

2.7 参考文献等

この章の内容のほとんどは [Uni13, 1 章、2 章] のものである。オリジナルの Martin-Löf 型理論については [ML75, ML82] を見よ。依存型理論には Martin-Löf のもの以外にも、pure type system [Bar92] や calculus of construction [CH88] がある。依存型を使ったプログラミングについては、[NPS90, Nor07] がある。

すべての型が ∞-グルーポイドであることの証明は Peter Lumsdaine [Lum10] と Benno van den Berg と Richard Garner [vdBG11] によって独立になされた。グルーポイドのファイブレーションについては、 [Bro70] を見よ。

3 Univalence Axiom と Higher Inductive Types

この章では HoTT の肝である univalence axiom と higher inductive type について述べる。

3.1 ホモトピーと同値

$f, g : \prod_{x:A} B(x)$ に対して、f から g への ホモトピー (homotopy) の型 $f \sim g : \mathcal{U}$ を

$$f \sim g \equiv \prod_{x:A} f(x) = g(x)$$

と定義する。$A, B : \mathcal{U}$ と $f : A \to B$ に対し,

$$\begin{aligned} \mathsf{linv}(f) &\equiv \sum_{g:B\to A} g \circ f \sim \mathsf{id} \\ \mathsf{rinv}(f) &\equiv \sum_{h:B\to A} f \circ h \sim \mathsf{id} \\ \mathsf{isequiv}(f) &\equiv \mathsf{linv}(f) \times \mathsf{rinv}(f) \end{aligned}$$

とし、

$$A \simeq B \equiv \sum_{f:A\to B} \mathsf{isequiv}(f)$$

と定義する。$\mathsf{isequiv}(f)$ の要素があるとき、f は 同値射 (equivalence) であるといい、$A \simeq B$ の要素があるとき、A と B は 同値 (equivalent) であるという。

注意 10. f が同値射であることの定義として、$\mathsf{isequiv}(f)$ の代わりに

$$\mathsf{qinv}(f) :\equiv \sum_{g:B\to A} (g \circ f \sim \mathsf{id}) \times (f \circ g \sim \mathsf{id})$$

を使う方が自然に思えるかもしれない。しかし、$\sum_{f:A\to B} \mathsf{qinv}(f)$ は「同値射の空間」として良く振る舞わないことが知られている。同値射の定義に関する議論については、 [Uni13, 4 章] を見よ。

例 11. 恒等射 $\mathsf{id}_A : A \to A$ は同値射である。同値射として見たときも同じ記号で $\mathsf{id}_A : A \simeq A$ と書く。

例 12. 整数全体のなす型を $\mathbb{Z} :\equiv \mathbb{N} + \mathbf{1} + \mathbb{N}$ と定義する。左の $\mathbb{N}$ が負の整数、右の $\mathbb{N}$ が正の整数、中央の $\mathbf{1}$ が 0 に対応する。$\mathsf{suc} : \mathbb{N} \to \mathbb{N}$ は自然に $\mathsf{suc} : \mathbb{Z} \to \mathbb{Z}$ に拡張され、$\mathsf{suc} : \mathbb{Z} \to \mathbb{Z}$ は同値射である。

3.2 Univalence Axiom

$\mathsf{idtoequiv} : \prod_{X,Y:\mathcal{U}} (X = Y) \to (X \simeq Y)$ を $\mathsf{idtoequiv}(X, X, \mathsf{refl}(X)) \equiv \mathsf{id}_X$ で定義する。Univalence axiom は $\mathsf{idtoequiv}$ が同値射であることを要請する公理である。

公理 13.

$$\mathsf{ua} : \prod_{X,Y:\mathcal{U}} \mathsf{isequiv}(\mathsf{idtoequiv}(X,Y))$$

Univalence axiom から直ちにわかることは、$X = Y$ と $X \simeq Y$ が同値だということである。これにより、同値射 $e : X \simeq Y$ はしばしば $X = Y$ の要素だとみなすことがある。

注意 14. Univalence axiom には次のような定式化もあるように思えるかもしれない。

1. $(X \simeq Y) \to (X = Y)$
2. $(X \simeq Y) \simeq (X = Y)$
3. $\mathsf{isequiv}(\mathsf{idtoequiv}(X,Y))$

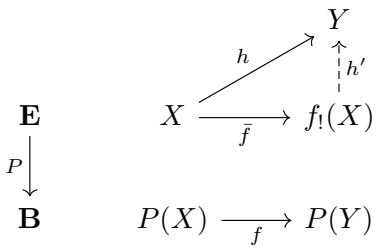
3 が公式の univalence axiom である。明らかに $3 \Rightarrow 2 \Rightarrow 1$ である。実は $2 \Rightarrow 3$ も成り立つが、$1 \Rightarrow 2$ が成り立つかどうかは分かっていない[*1]。

3.3 依存道

Higher inductive type を導入する準備として、 依存道 (dependent path) について述べる。

$A : \mathcal{U}, B : A \to U, x : A, y : A, p : x = y, u : B(x), v : B(y)$ とする。u から v への p 上の道 (path lying over p) とは、$B(y)$ の道 $q : \mathsf{transport}(B,p,u) = v$ のことである。$u =^B_p v :\equiv \mathsf{transport}(B,p,u) = v$ と書く。他の道上の道は一般に 依存道 と呼ばれる。

注意 15. $q : \mathsf{transport}(B,p,u) = v$ を p 上の道と呼ぶのにはグルーポイドのファイブレーションとの類似がある。$P : \mathbf{E} \to \mathbf{B}$ をグルーポイドのファイブレーション、$X, Y \in \mathbf{E}$, $f : P(X) \to P(Y)$ とする。このとき、 f 上の射 $h : X \to Y$ に対して、$P(Y)$ 上の射 $h' : f_!(X) \to Y$ で $h' \circ \bar{f} = h$ となるものがただ一つ存在する。



このことにより、 $P(Y)$ 上の射 $h' : f_!(X) \to Y$ を X から Y への f 上の射と思うことができ、射 $f_!(X) \to Y$ に対応するのが道 $\mathsf{transport}(B,p,u) = v$ である。

ap の依存関数版を考える。$A : \mathcal{U}, B : A \to \mathcal{U}$ のとき、

$$\mathsf{apd} : \prod_{f:\prod_{x:A} B(x)} \prod_{x:A,y:A,p:x=y} f(x) =^B_p f(y)$$

$$\mathsf{apd}(f, \mathsf{refl}(x)) :\equiv \mathsf{refl}(f(x))$$

[*1] https://groups.google.com/forum/#!topic/homotopytypetheory/aTx_oXjA5qA

とする。$(u =^{B}_{\mathsf{refl}(x)} v) \equiv (u = v)$ であることに注意。ap のときと同様に、単に $f(p) :\equiv \mathsf{apd}(f, p)$ と書くこともある。

3.4 Higher Inductive Types

Higher inductive type は帰納的型の拡張で、従来の構成子に加えて 道構成子 (path constructor) を持つ。違いをはっきりさせるために、従来の構成子は 点構成子 (point constructor) と呼ぶ。

3.4.1 区間

区間 (interval) I は次で定義される。

1. 構成規則
   $I : \mathcal{U}$
2. 構成子
   (a) 点構成子 $0 : I$
   (b) 点構成子 $1 : I$
   (c) 道構成子 $\mathsf{seg} : 0 = 1$
3. 除去子
   $\mathsf{ind}_I : \prod_{P:I\to\mathcal{U}} \prod_{c:P(0)} \prod_{d:P(1)} c =^{P}_{\mathsf{seg}} d \to \prod_{x:I} P(x)$
4. 計算規則
   (a) $\mathsf{ind}_I(P, c, d, s, 0) \equiv c$
   (b) $\mathsf{ind}_I(P, c, d, s, 1) \equiv d$
   (c) $\mathsf{ind}_I{-}\mathsf{seg}{-}\beta : \mathsf{ind}_I(P, c, d, s, \mathsf{seg}) = s$

構成子からわかるように、区間 I は二つの端点 0 と 1 と、その間を結ぶ道 $\mathsf{seg} : 0 = 1$ で生成される空間である。除去子が意味するところは、 I からの関数 $f : \prod_{x:I} P(x)$ を作るには、端点 $0, 1$ の行き先 $f(0) : P(0), f(1) : P(1)$ と seg の行き先 $\mathsf{apd}(f, \mathsf{seg}) : f(0) =^{P}_{\mathsf{seg}} f(1)$ を決めれば十分だということである。計算規則には少し注意がいる。点構成子についての計算規則は要素の等しさ $\equiv$ で定める一方、道構成子についての計算規則は等式型の要素 $\mathsf{ind}_I{-}\mathsf{seg}{-}\beta$ を追加することで定めている。道に対する作用 $\mathsf{apd}(\mathsf{ind}_I(P, c, d, s))$ はプリミティブではなく我々が定義したものなので、$\mathsf{apd}(\mathsf{ind}_I(P, c, d, s), \mathsf{seg}) \equiv s$ とする正当性は無いのである。

3.4.2 $\mathbb{S}^1$

円周 (circle) $\mathbb{S}^1$ は次で定義される。

1. 構成規則
   $\mathbb{S}^1 : \mathcal{U}$
2. 構成子

(a) 点構成子 $\mathsf{base} : \mathbb{S}^1$

(b) 道構成子 $\mathsf{loop} : \mathsf{base} = \mathsf{base}$

3. 除去子

$\mathsf{ind}_{\mathbb{S}^1} : \prod_{P:\mathbb{S}^1 \to \mathcal{U}} \prod_{b:P(\mathsf{base})} b =^{P}_{\mathsf{loop}} b \to \prod_{x:\mathbb{S}^1} P(x)$

4. 計算規則

(a) $\mathsf{ind}_{\mathbb{S}^1}(P, b, l, \mathsf{base}) \equiv b$

(b) $\mathsf{ind}_{\mathbb{S}^1} - \mathsf{loop} - \beta : \mathsf{ind}_{\mathbb{S}^1}(P, b, l, \mathsf{loop}) = l$

4 円周の基本群

次を示す。

定理 16.

$$(\mathsf{base} = \mathsf{base}) \simeq \mathbb{Z}$$

定理 16 は $\mathbb{S}^1$ の base 上のループ空間 (loop space) が $\mathbb{Z}$ と同値であることを意味する。HoTT での基本群の定義は本チュートリアルでは割愛するが、定理 16 から直ちに $\mathbb{S}^1$ の基本群が $\mathbb{Z}$ であることが従う。

さて、定理の証明に入るが、$\mathbb{S}^1$ の帰納法を使うために、一般化された命題を考える。まず、$\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ であって $\mathsf{code}(\mathsf{base}) \equiv \mathbb{Z}$ である型の族を定義する。そして、

$$\prod_{x:\mathbb{S}^1} (\mathsf{base} = x) \simeq \mathsf{code}(x)$$

を証明する。$x :\equiv \mathsf{base}$ とすれば定理 16 を得る。

$\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ を帰納法で定義する。

$$\mathsf{code}(\mathsf{base}) \equiv \mathbb{Z}$$
$$\mathsf{code}(\mathsf{loop}) = \mathsf{suc}$$

ここで、$\mathsf{suc} : \mathbb{Z} \simeq \mathbb{Z}$ であるが、 univalence により $\mathsf{suc} : \mathbb{Z} = \mathbb{Z}$ とみなしている。

$\mathsf{encode} : \prod_{x:\mathbb{S}^1} \mathsf{base} = x \to \mathsf{code}(x)$ と $\mathsf{decode} : \prod_{x:\mathbb{S}^1} \mathsf{code}(x) \to \mathsf{base} = x$ を帰納法で定義する。encode は道帰納法により

$$\mathsf{encode}(\mathsf{base}, \mathsf{refl}(\mathsf{base})) :\equiv 0$$

とすればよい。decode は $\mathbb{S}^1$ の帰納法で定義する。$\mathsf{decode}(\mathsf{base}) : \mathbb{Z} \to \mathsf{base} = \mathsf{base}$ は

$$\mathsf{decode}(\mathsf{base}, n) :\equiv \mathsf{loop}^n$$

とする。ここで、 loop^n は loop を n 個結合した道である。後は

$$\mathsf{decode}(\mathsf{loop}) : \mathsf{decode}(\mathsf{base}) =^{\lambda x.\mathsf{code}(x) \to \mathsf{base} = x}_{\mathsf{loop}} \mathsf{decode}(\mathsf{base})$$

を与えなければならないが、これの型は次の図式の可換性と同値であることが示せる。

$$\begin{array}{ccc} \mathbb{Z} & \xrightarrow{\mathsf{decode}(\mathsf{base})} & \mathsf{base} = \mathsf{base} \\ {\scriptstyle \mathsf{suc}}\downarrow & & \downarrow{\scriptstyle \lambda p.p\cdot\mathsf{loop}} \\ \mathbb{Z} & \xrightarrow[\mathsf{decode}(\mathsf{base})]{} & \mathsf{base} = \mathsf{base} \end{array}$$

$\mathsf{decode}(\mathsf{base}, n) \equiv \mathsf{loop}^n$ なのでこの図式は可換である。この可換性の証明を先の同値で引き戻したものを $\mathsf{decode}(\mathsf{loop})$ とする。

補題 17. *1.* $\prod_{x:\mathbb{S}^1} \mathsf{decode}(x) \circ \mathsf{encode}(x) \sim \mathsf{id}_{\mathsf{base}=x}$

2. $\prod_{x:\mathbb{S}^1} \mathsf{encode}(x) \circ \mathsf{decode}(x) \sim \mathsf{id}_{\mathsf{code}(x)}$

Proof. 1 は簡単な道帰納法で示せる。2 はやや複雑だが、$\mathbb{S}^1$ の帰納法で示せる。 □

補題 17 より、$\prod_{x:\mathbb{S}^1}(\mathsf{base} = x) \simeq \mathsf{code}(x)$ が示され、$x \equiv \mathsf{base}$ として

$$(\mathsf{base} = \mathsf{base}) \simeq \mathbb{Z}$$

を得る。

ここで使った証明手法は encode-decode 法 (encode-decode method) と呼ばれる、ある型の等式型の特徴付けによく使われる手法である。等式型をより詳細に記述する code という型の族を定義し、等式型と code の間の同値射を作るというのが大まかなやり方である。帰納法を使いやすいスタイルであり、ホモトピー論に対する「型理論的」なアプローチと言える。

4.1 参考文献等

HoTT での円周の基本群の計算は Licata と Shulman [LS13] によるものである。従来の数学では、円周の基本群の計算には 普遍被覆 (universal cover) を使う (例えば [Hat01] を見よ)。HoTT での証明でも、実は $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ が $\mathbb{S}^1$ 上の普遍被覆を与えるが、encode-decode 法ではその事実は使わない。

5 さらに読む

HoTT の研究は大きく分けて二つの方向が考えられる。一つは HoTT を使って代数トポロジーや数学を展開し、定理証明器上で実装することである。HoTT の基本的なライブラリは Coq [BGL+17] や Agda[*2] で開発されている他、Lean 2[*3] も HoTT をサポートしている。個別のトピックについては、円周の基本群 [LS13]、ホモトピー極限 [AKL15]、Eilenberg-MacLane 空間 [LF14]、3 次元球面の 4 次ホモトピー群 [Bru16]、Blakers-Massey 定理 [HFFLL16]、Seifert-van Kampen 定理 [HS16] 等がある。

[*2] https://github.com/HoTT/HoTT-Agda

[*3] https://github.com/leanprover/lean2

もう一つは、 HoTT のメタ理論を研究することである。最も大きな関心は、 univalence axiom と HIT の性質である。期待されることは、 univalence と HIT は Martin-Löf 型理論の性質を大きくは変えないということで、例えば Voevodsky の homotopy canonicity 予想 がある。

予想 18. *HoTT* で定義できる $\mathbb{N}$ 型の閉項 $t : \mathbb{N}$ に対し、ある自然数 n が存在し、$t = \mathsf{suc}^n(0)$ を *HoTT* で証明できる。

Homotopy canonicity 予想については、Shulman [Shu15] が gluing construction を使って部分的に解決している。また、 Johann と Sojakova [JS17] はhigher dimensional parametricity の概念が homotopy canonicity 予想の解決に有効であることを仄めかしている。

定理証明器で形式化する際に重要となるのは、univalence と HIT の計算的意味 (computational meaning) である。Univalence の定義を思い出すと、定数項 ua を追加しただけであり、ua を含む項がどう簡約されるかという規則は定められていない。項を自動で簡約できることは定理証明器の大きな利点の一つであるから、univalence と HIT によってこれが破壊されないことが望ましい。Univalence と HIT の計算的意味へのアプローチとして、cubical type theory (CTT) [CCHM16] がある。これは cubical set の圏が HoTT のモデルになることを構成的数学で証明できる [BCH14] ことにヒントを得て設計された型理論で、 CTT では univalence axiom を証明できる。また、実装もされている[*4]。

もう少し抽象的な問いとして、HoTT のモデルは何かというものがある。依存型理論のモデルとしては、 contextual category [Car86], category with families [Dyb96], comprehension category [Jac93] 等がある。ホモトピー論で使われるモデル圏 [Qui67] や fibration category [Bro73] に近いものには、logical model category [AK11] やtype-theoretic fibration category [Shu15] がある。特に type-theoretic fibration category は健全かつ完全なモデルである。高次圏論を使ったモデルも関心が高い。HoTT は ∞-topos [Lur09] の内部言語 (internal language) であると予想されている。この予想の詳細は例えば [Kap15, KL16] を見よ。

全く別の方向として、計算機科学や数学の他の分野への HoTT の応用もある。Homotopical patch theory [AMLH14] は HoTT をバージョン管理システムに応用したもので、差分を道と思うことが基本的なアイディアである。

参考文献

[AK11] Peter Arndt and Krzysztof Kapulkin. Homotopy-Theoretic Models of Type Theory. In Luke Ong, editor, Typed Lambda Calculi and Applications: 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings, pages 45–60, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-21691-6_7.

[*4] https://github.com/mortberg/cubicaltt

[AKL15] Jeremy Avigad, Krzysztof Kapulkin, and Peter LeFanu Lumsdaine. Homotopy limits in type theory. Mathematical Structures in Computer Science, 25(5):1040 – 1070, 2015. arXiv:1304.0680v3, doi:10.1017/S0960129514000498.

[AMLH14] Carlo Angiuli, Edward Morehouse, Daniel R. Licata, and Robert Harper. Homotopical Patch Theory. In Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming, ICFP '14, pages 243–256, New York, NY, USA, 2014. ACM. doi:10.1145/2628136.2628158.

[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. Mathematical Proceedings of the Cambridge Philosophical Society, 146(1):45 – 55, 2009. doi:10.1017/S0305004108001783.

[Bar92] H. P. Barendregt. Lambda Calculi with Types. In S. Abramsky, Dov M. Gabbay, and S. E. Maibaum, editors, Handbook of Logic in Computer Science (Vol. 2), pages 117–309. Oxford University Press, Inc., New York, NY, USA, 1992.

[BCH14] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In Ralph Matthes and Aleksy Schubert, editors, 19th International Conference on Types for Proofs and Programs (TYPES 2013), volume 26 of Leibniz International Proceedings in Informatics (LIPIcs), pages 107–128, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2013.107.

[BGL+17] Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, Matthieu Sozeau, and Bas Spitters. The HoTT Library: A Formalization of Homotopy Type Theory in Coq. In Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, pages 164–172, New York, NY, USA, 2017. ACM. arXiv:1610.04591, doi:10.1145/3018610.3018615.

[Bro70] Ronald Brown. Fibrations of Groupoids. Journal of Algebra, 15(1):103 – 132, 1970. doi:10.1016/0021-8693(70)90089-X.

[Bro73] Kenneth S. Brown. Abstract homotopy theory and generalized sheaf cohomology. Transactions of the American Mathematical Society, 186:419–458, 1973. doi:10.2307/1996573.

[Bru16] Guillaume Brunerie. On the homotopy groups of spheres in homotopy type theory. PhD thesis, University of Nice, 2016. arXiv:1606.05916v1.

[Car86] John Cartmell. Generalised algebraic theories and contextual categories. Annals of Pure and Applied Logic, 32:209 – 243, 1986. doi:10.1016/0168-0072(86)90053-9.

[CCHM16] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: a constructive interpretation of the univalence axiom. 2016. arXiv:1611.02108v1.

[CH88] Thierry Coquand and Gérard Huet. The Calculus of Constructions. Information and Computation, 76(2):95 – 120, 1988. doi:10.1016/0890-5401(88)90005-3.

[Dyb96] Peter Dybjer. Internal Type Theory. In Stefano Berardi and Mario Coppo, editors, Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers, pages 120–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. doi:10.1007/3-540-61780-9_66.

[Hat01] Allen Hatcher. Algebraic Topology. Cambridge University Press, 2001. URL: https://www.math.cornell.edu/~hatcher/AT/ATpage.html.

[HFFLL16] Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. A Mechanization of the Blakers-Massey Connectivity Theorem in Homotopy Type Theory. In Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, pages 565–574, New York, NY, USA, 2016. ACM. arXiv:1605.03227v1, doi:10.1145/2933575.2934545.

[HS98] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In Twenty-five years of constructive type theory (Venice, 1995), volume 36 of Oxford Logic Guides, pages 83–111. Oxford Univ. Press, New York, 1998.

[HS16] Kuen-Bang Hou (Favonia) and Michael Shulman. The Seifert-van Kampen Theorem in Homotopy Type Theory. In Jean-Marc Talbot and Laurent Regnier, editors, 25th EACSL Annual Conference on Computer Science Logic (CSL 2016), volume 62 of Leibniz International Proceedings in Informatics (LIPIcs), pages 22:1–22:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2016.22.

[Jac93] Bart Jacobs. Comprehension categories and the semantics of type dependency. Theoretical Computer Science, 107(2):169 – 207, 1993. doi:10.1016/0304-3975(93)90169-T.

[JS17] Patricia Johann and Kristina Sojakova. Cubical Categories for Higher-Dimensional Parametricity. January 2017. URL: http://arxiv.org/abs/1701.06244v1, arXiv:1701.06244v1.

[Kap15] Chris Kapulkin. Locally Cartesian Closed Quasicategories from Type Theory. July 2015. URL: http://arxiv.org/abs/1507.02648v1, arXiv:1507.02648v1.

[KL16] Chris Kapulkin and Peter LeFanu Lumsdaine. The homotopy theory of type theories. 2016. arXiv:1610.00037v1.

[KLV12] Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. Univalence in Simplicial Sets. 2012. arXiv:1203.2553v3.

[LF14] Daniel R. Licata and Eric Finster. Eilenberg-MacLane Spaces in Homotopy Type Theory. In Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual

ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, pages 66:1–66:9, New York, NY, USA, 2014. ACM. doi:10.1145/2603088.2603153.

[LS13] Daniel R. Licata and Michael Shulman. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '13, pages 223–232, Washington, DC, USA, 2013. IEEE Computer Society. arXiv:1301.3443v1, doi:10.1109/LICS.2013.28.

[Lum10] Peter Lumsdaine. Weak omega-categories from intensional type theory. Logical Methods in Computer Science, 6(3), September 2010. doi:10.2168/lmcs-6(3:24)2010.

[Lur09] Jacob Lurie. Higher Topos Theory. Princeton University Press, 2009.

[ML75] Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. Studies in Logic and the Foundations of Mathematics, 80:73 – 118, 1975. doi:10.1016/S0049-237X(08)71945-1.

[ML82] Per Martin-Löf. Constructive Mathematics and Computer Programming. Studies in Logic and the Foundations of Mathematics, 104:153 – 175, 1982. doi:10.1016/S0049-237X(09)70189-2.

[Nor07] Ulf Norell. Towards a practical programming language based on dependent type theory. PhD thesis, Chalmers, Göteborg University, 2007. URL: http://www.cse.chalmers.se/~ulfn/papers/thesis.pdf.

[NPS90] Bengt Nordström, Kent Petersson, and Jan M. Smith. Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press, 1990. URL: http://www.cse.chalmers.se/research/group/logic/book/.

[PW14] Álvaro Pelayo and Michael Warren. Homotopy type theory and Voevodsky's univalent foundations. Bulletin of the American Mathematical Society, 51(4):597–648, 2014. arXiv:1210.5658v1, doi:10.1090/s0273-0979-2014-01456-9.

[Qui67] Daniel G. Quillen. Homotopical Algebra. Springer, 1967.

[Shu15] Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. Mathematical Structures in Computer Science, 25(05):1203–1277, 2015. arXiv:1203.3253v3, doi:10.1017/s0960129514000565.

[Uni13] Univalent Foundations Program. Homotopy Type Theory: Univalent Foundations of Mathematics Institute for Advanced Study, 2013. URL: http://homotopytypetheory.org/book/.

[vdBG11] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. Proceedings of the London Mathematical Society, 102(2):370–394, 2011. doi:

10.1112/plms/pdq026.

[Voe06] Vladimir Voevodsky. A very short note on homotopy λ-calculus. September 2006. URL: http://www.math.ias.edu/vladimir/files/2006_09_Hlambda.pdf.

[Voe09] Vladimir Voevodsky. Notes on type systems. September 2009. URL: https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/expressions_current.pdf.

[Voe10] Vladimir Voevodsky. Univalent Foundations Project. a modified version of an NSF grant application, October 2010. URL: http://www.math.ias.edu/vladimir/files/univalent_foundations_project.pdf.

[War08] Michael A. Warren. Homotopy Theoretic Aspects of Constructive Type Theory. PhD thesis, Carnegie Mellon University, 2008. URL: http://mawarren.net/papers/phd.pdf.