Issue XXXVII: Simplicial Type Theory

Maksym Sokhatskyi [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnical Institute
15 травня 2025 р.

Анотація

We propose a synthetic framework for simplicial homotopy theory within homotopy type theory, axiomatizing a directed interval type to define higher simplices and probe the simplicial structure of types. We introduce *Segal types*, where binary composites are unique up to homotopy, ensuring coherent associativity; *Rezk types*, where categorical isomorphisms coincide with type-theoretic identities; and *Kan types*, satisfying a horn-filling condition modeling ∞-groupoids. We define covariant fibrations as functorial type families and prove a dependent Yoneda lemma, providing a directed analogue of identity elimination. Semantically, our types correspond to Segal spaces, complete Segal spaces, and Kan complexes in bisimplicial sets, offering a synthetic language for simplicial homotopy theory.

Keywords: Simplicial Homotopy Theory

Зміст

1 Simplicial Homotopy Type Theory

Homotopy type theory (HoTT) [1] extends Martin-Löf type theory with axioms, such as the univalence axiom, enabling it to serve as a synthetic language for ∞-groupoids, as modeled by simplicial sets in Voevodsky's model [2]. This paper develops a synthetic simplicial homotopy theory within HoTT, focusing on three flavors: Kan complexes (modeling ∞-groupoids), Segal spaces (modeling weak categories), and complete Segal spaces or quasi-categories (modeling $(\infty, 1)$-categories).

In standard HoTT, types are synthetic ∞-groupoids, with identity types providing paths and higher homotopies. However, simplicial homotopy theory requires richer structures, such as directed arrows and compositions, as in Segal or Rezk spaces. Following [3], we interpret HoTT in the Reedy model structure on bisimplicial sets, where types are simplicial spaces, and identify specific types—Segal, Rezk, and Kan types—corresponding to Segal spaces, complete Segal spaces, and Kan complexes, respectively.

Our approach axiomatizes a directed interval type $\mathbb{2}$, a strict totally ordered set with endpoints $0, 1 : \mathbb{2}$, modeled by the simplicial 1-simplex Δ^1 in the categorical direction of bisimplicial sets [4]. This allows us to define higher simplices, e.g., $\Delta^2 = \{(s, t) : \mathbb{2} \times \mathbb{2} \mid t \leq s\}$, and probe the simplicial structure of types via maps $\Delta^n \to A$.

We define:

- *Segal types*, where composable arrows have contractible spaces of composites, ensuring categorical coherence.
- *Rezk types*, Segal types where isomorphisms are equivalent to identities, modeling quasi-categories.
- *Kan types*, where horn inclusions $\Lambda^n_i \to \Delta^n$ lift uniquely up to homotopy, modeling ∞-groupoids.

We study functors (type-theoretic functions), natural transformations ($A \times \mathbb{2} \to B$), and covariant fibrations (functorial type families), proving a dependent Yoneda lemma. In Appendix A, we show that our types correspond to their semantic counterparts in bisimplicial sets, leveraging the Reedy and Rezk model structures.

This synthetic framework simplifies reasoning about simplicial homotopy theory, as type-theoretic operations are automatically functorial, mirroring the internalization benefits of [3].

1.1 Simplicial Types

We extend HoTT with a strict interval type $\mathbb{2}$, a totally ordered set with distinct elements $0, 1 : \mathbb{2}$, satisfying the coherent theory of a strict interval [5, 6]. Semantically, $\mathbb{2}$ is the simplicial 1-simplex Δ^1 in the categorical direction of bisimplicial sets.

Definition 1. The *strict interval* $\mathbb{2}$ is equipped with:

$$\begin{cases} 0, 1 : \mathbb{2}, & \text{distinct endpoints,} \\ \leq : \mathbb{2} \times \mathbb{2} \to \mathcal{U}, & \text{total order, with } 0 \leq 1. \end{cases}$$

Higher simplices are defined internally:

$$\Delta^n = \{(s_1, \ldots, s_n) : \mathbb{2}^n \mid s_1 \leq s_2 \leq \cdots \leq s_n\}.$$

For example, $\Delta^2 = \{(s, t) : \mathbb{2} \times \mathbb{2} \mid t \leq s\}$. A map $\alpha : \Delta^2 \to A$ represents a commutative triangle in A, with edges $\lambda t.\alpha(t, 0)$, $\lambda t.\alpha(1, t)$, and $\lambda t.\alpha(t, t)$.

We use extension types to define hom-types. For $x, y : A$, the type of arrows from x to y is:

$$\hom_A(x, y) := \left\langle \prod_{f:\mathbb{2}\to A} \mathcal{U} \mid \{0, 1\} \hookrightarrow \mathbb{2} \right\rangle \{f(0) = x, f(1) = y\},$$

where $\{0, 1\} \hookrightarrow \mathbb{2}$ is a cofibration, and the type family enforces $f(0) \equiv x$, $f(1) \equiv y$ judgmentally, avoiding identity type data [3].

1.1.1 Segal Types

Segal types model synthetic Segal spaces, where composition is unique up to homotopy.

Definition 2. A type A is a *Segal type* if, for any composable arrows $f, g : \mathbb{2} \to A$ with $f(1) = g(0)$, the type of composites

$$\sum_{h:\mathbb{2}\to A} \sum_{\alpha:\mathbb{2}\to A} (\lambda t.\alpha(t, 0) = f) \times (\lambda t.\alpha(1, t) = g) \times (\lambda t.\alpha(t, t) = h)$$

is contractible.

This ensures that composition is associative and unital up to all higher homotopies, as the contractibility condition implies the Segal map $A^{\Delta^2} \to A^{\Delta^1} \times_{A^{\Delta^0}} A^{\Delta^1}$ is an equivalence in bisimplicial sets.

Theorem 1. In a Segal type A, composition is coherently associative and unital.

Доведення. The contractibility of the composite type implies that the Segal maps for higher simplices (e.g., $A^{\Delta^3} \to A^{\Delta^1} \times_{A^{\Delta^0}} A^{\Delta^1} \times_{A^{\Delta^0}} A^{\Delta^1}$) are equivalences, ensuring associativity and unit laws hold up to homotopy, as in [7]. □

1.1.2 Rezk Types

Rezk types model complete Segal spaces, where isomorphisms coincide with identities.

Definition 3. A Segal type A is a *Rezk type* if the type of isomorphisms

$$\mathrm{iso}_A(x, y) :\equiv \sum_{f:\mathrm{hom}_A(x,y)} \sum_{g:\mathrm{hom}_A(y,x)} (\text{inverses up to homotopy})$$

is equivalent to the identity type $x = y$.

This "local univalence" condition ensures A models a quasi-category, where invertible arrows are precisely paths.

Theorem 2. Rezk types correspond to complete Segal spaces in the Rezk model structure on bisimplicial sets.

Доведення. The completeness condition corresponds to the map $A^{E\square\Delta^0} \to A^{\Delta^0\square\Delta^0}$ being a trivial fibration, as in [3], matching Definition A.24 of the original paper. □

1.1.3 Kan Types

Kan types model synthetic Kan complexes, satisfying a horn-filling condition.

Definition 4. A type A is a *Kan type* if, for all $n \geq 1$ and $0 \leq i \leq n$, the horn inclusion $\Lambda^n_i \to \Delta^n$ induces a contractible type of fillers:

$$\sum_{f:\Delta^n \to A} \sum_{g:\Lambda^n_i \to A} (g = f \circ \iota)\,.$$

Kan types model ∞-groupoids, as every horn has a unique filler up to homotopy, corresponding to Kan complexes in simplicial sets.

Theorem 3. Kan types are Segal types, and every Kan type is a Rezk type.

Доведення. The Kan condition implies the Segal condition, as horn-filling for Λ^2_1 ensures unique composites. The Kan condition also implies all arrows are invertible, satisfying the Rezk completeness condition trivially. □

1.2 Covariant Fibrations and the Yoneda Lemma

We define covariant fibrations as functorial type families over Segal types.

Definition 5. A type family $C : A \to \mathcal{U}$ over a Segal type A is a *covariant fibration* if, for any $a : A$, $C(a)$ is a Kan type, and for any $f : \hom_A(a, b)$, there is a transport map $C(f) : C(a) \to C(b)$ satisfying functoriality up to homotopy.

Theorem 4 (Dependent Yoneda Lemma)**.** For a covariant fibration $C : A \to \mathcal{U}$ and $a : A$, there is an equivalence

$$C(a) \simeq \hom_{\mathrm{Fib}_A}(y(a), C),$$

where $y(a) : A \to \mathcal{U}$, $y(a)(b) = \hom_A(a, b)$, is the Yoneda embedding, and Fib_A is the type of covariant fibrations over A.

Доведення. The proof follows [3], constructing a natural equivalence via the contractibility of hom-types and functoriality of C, internalized in the type theory. □

1.3 Synthetic Categorical Structures

We extend the framework to include synthetic analogues of categorical structures, such as natural transformations, adjunctions, limits, and discrete types, which enrich the simplicial homotopy theory.

Definition 6. For Segal types A, B and functors $F, G : A \to B$ (i.e., type-theoretic functions preserving Segal structure), a *natural transformation* $\eta : F \to G$ is a map

$$\eta : \prod_{a:A} \hom_B(F(a), G(a)),$$

such that for any $f : \hom_A(a, a')$, the following diagram commutes up to homotopy:

$$\begin{array}{ccc} F(a) & \xrightarrow{\eta_a} & G(a) \\ \downarrow{\scriptstyle F(f)} & & \downarrow{\scriptstyle G(f)} \\ F(a') & \xrightarrow{\eta_{a'}} & G(a') \end{array}$$

Theorem 5. For Segal types A, B, the type of natural transformations $\prod_{F,G:A\to B} \prod_{a:A} \hom_B(F(a), G(a))$ is a Segal type.

Доведення. The naturality condition ensures that the type of transformations satisfies the Segal condition, as the hom-types $\hom_B(F(a), G(a))$ are contractible for composable arrows, and the functoriality of F, G preserves this structure, following [7]. □

Definition 7. For Segal types A, B, an *adjunction* consists of functors $\mathsf{F} : \mathsf{A} \to \mathsf{B}$, $\mathsf{G} : \mathsf{B} \to \mathsf{A}$, and natural transformations

$$\begin{cases} \eta : \mathrm{id}_{\mathsf{A}} \to \mathsf{G} \circ \mathsf{F}, & \text{unit}, \\ \epsilon : \mathsf{F} \circ \mathsf{G} \to \mathrm{id}_{\mathsf{B}}, & \text{counit}, \end{cases}$$

satisfying the triangle identities up to homotopy:

$$(\mathsf{G}\epsilon) \circ (\eta\mathsf{G}) \simeq \mathrm{id}_{\mathsf{G}}, \quad (\epsilon\mathsf{F}) \circ (\mathsf{F}\eta) \simeq \mathrm{id}_{\mathsf{F}}.$$

Theorem 6. An adjunction $(\mathsf{F}, \mathsf{G}, \eta, \epsilon)$ between Segal types A, B induces an equivalence

$$\hom_{\mathsf{B}}(\mathsf{F}(\mathfrak{a}), \mathfrak{b}) \simeq \hom_{\mathsf{A}}(\mathfrak{a}, \mathsf{G}(\mathfrak{b}))$$

for all $\mathfrak{a} : \mathsf{A}$, $\mathfrak{b} : \mathsf{B}$.

Доведення. The unit and counit induce a bijection on hom-types via the triangle identities, which holds up to homotopy in the Segal type structure, mirroring the categorical adjunction in [4]. □

Definition 8. For a Segal type A and a diagram $\mathsf{D} : \mathsf{I} \to \mathsf{A}$ (where I is a Segal type), a *limit* of D is a Kan type L with a natural transformation $\pi : \mathrm{const}_{\mathsf{L}} \to \mathsf{D}$ such that, for any Kan type X and natural transformation $\sigma : \mathrm{const}_{\mathsf{X}} \to \mathsf{D}$, there exists a unique map $\mathsf{f} : \mathsf{X} \to \mathsf{L}$ with $\pi \circ \mathrm{const}_{\mathsf{f}} \simeq \sigma$.

Theorem 7. In a Rezk type A, the limit of any diagram $\mathsf{D} : \mathsf{I} \to \mathsf{A}$ is a Kan type.

Доведення. The limit L inherits the Kan condition from the fibers of the projection π, as Rezk types ensure isomorphisms are identities, and the universal property enforces contractibility of the mapping space, as in [3]. □

Definition 9. A type A is *discrete* if its identity types $\mathfrak{a} =_{\mathsf{A}} \mathfrak{b}$ are propositions (0-truncated), i.e., for all $\mathfrak{a}, \mathfrak{b} : \mathsf{A}$ and $\mathfrak{p}, \mathfrak{q} : \mathfrak{a} =_{\mathsf{A}} \mathfrak{b}$, we have $\mathfrak{p} = \mathfrak{q}$.

Theorem 8. For any discrete type A, there exists a Segal type $\hat{\mathsf{A}}$ and an embedding $\mathfrak{i} : \mathsf{A} \to \hat{\mathsf{A}}$ such that $\hom_{\hat{\mathsf{A}}}(\mathfrak{i}(\mathfrak{a}), \mathfrak{i}(\mathfrak{b})) \simeq (\mathfrak{a} =_{\mathsf{A}} \mathfrak{b})$.

Доведення. Construct $\hat{\mathsf{A}}$ as the Segal type generated by A with hom-types $\hom_{\hat{\mathsf{A}}}(\mathfrak{i}(\mathfrak{a}), \mathfrak{i}(\mathfrak{b})) := (\mathfrak{a} =_{\mathsf{A}} \mathfrak{b})$, which is contractible for $\mathfrak{a} = \mathfrak{b}$ and empty otherwise, satisfying the Segal condition and embedding A via the Yoneda lemma, as in [2]. □

1.4 Synthetic ∞-categories

1.4.1 Strict Interval

Definition 10 (Interval Formation)**.** The strict interval type is formed as:

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash \mathbf{2} : \mathcal{U}}$$

```
def Interval_form : U := 2
```

Definition 11 (Interval Introduction)**.** Endpoints and order are introduced:

$$\frac{\Gamma \text{ ctx}}{\Gamma \vdash 0 : \mathbf{2}} \quad \frac{\Gamma \text{ ctx}}{\Gamma \vdash 1 : \mathbf{2}} \quad \frac{\Gamma \vdash s, t : \mathbf{2}}{\Gamma \vdash s \leq t : \mathcal{U}}$$

```
def Interval_intro_0 : Interval_form := 0
def Interval_intro_1 : Interval_form := 1
def Interval_order (s t: Interval_form) : U := s ≤ t
```

Definition 12 (Interval Elimination)**.** The interval is eliminated by case analysis:

$$\frac{\Gamma \vdash a : \mathbf{2} \quad \Gamma \vdash C : \mathbf{2} \to \mathcal{U} \quad \Gamma \vdash c_0 : C(0) \quad \Gamma \vdash c_1 : C(1)}{\Gamma \vdash \text{ind}_{\mathbf{2}}(a, \lambda x.C(x), c_0, c_1) : C(a)}$$

```
def Interval_elim (C: Interval_form -> U)
    (c0: C Interval_intro_0) (c1: C Interval_intro_1) (a: Interval_form)
  : C a := split { 0 -> c0 | 1 -> c1 }
```

Theorem 9 (Interval Computation)**.** Elimination reduces on endpoints:

$$\frac{\Gamma \vdash C : \mathbf{2} \to \mathcal{U} \quad \Gamma \vdash c_0 : C(0) \quad \Gamma \vdash c_1 : C(1)}{\Gamma \vdash \text{ind}_{\mathbf{2}}(0, \lambda x.C(x), c_0, c_1) \equiv c_0 : C(0)}$$

$$\frac{\Gamma \vdash C : \mathbf{2} \to \mathcal{U} \quad \Gamma \vdash c_0 : C(0) \quad \Gamma \vdash c_1 : C(1)}{\Gamma \vdash \text{ind}_{\mathbf{2}}(1, \lambda x.C(x), c_0, c_1) \equiv c_1 : C(1)}$$

```
def Interval_comp_0 (C: Interval_form -> U)
    (c0: C Interval_intro_0) (c1: C Interval_intro_1)
  : Ξ (C Interval_intro_0)
      (Interval_elim C c0 c1 Interval_intro_0) c0
 := refl (C Interval_intro_0) c0

def Interval_comp_1 (C: Interval_form -> U)
    (c0: C Interval_intro_0) (c1: C Interval_intro_1)
  : Ξ (C Interval_intro_1)
      (Interval_elim C c0 c1 Interval_intro_1) c1
 := refl (C Interval_intro_1) c1
```

Theorem 10 (Interval Uniqueness)**.** Elimination is unique for dependent types:

$$\frac{\Gamma \vdash a : \mathbf{2} \quad \Gamma \vdash C : \mathbf{2} \to \mathcal{U} \quad \Gamma \vdash c : C(a)}{\Gamma \vdash c \equiv \mathrm{ind}_{\mathbf{2}}(a, \lambda x.C(x), c[0/x], c[1/x]) : C(a)}$$

```
def Interval_uniq (C: Interval_form -> U) (a: Interval_form) (c: C a)
  : Ξ (C a) c (Interval_elim C (c[0/x]) (c[1/x]) a)
 := refl (C a) c
```

1.4.2 Shape Cubes

Definition 13 (Cube Formation)**.** Cube types are formed for shapes:

$$\frac{\Xi \text{ cube ctx}}{\Xi \vdash I : \text{Cube}} \quad \frac{\Xi \text{ cube ctx}}{\Xi \vdash 1 : \text{Cube}} \quad \frac{\Xi \vdash I : \text{Cube} \quad \Xi \vdash J : \text{Cube}}{\Xi \vdash I \times J : \text{Cube}}$$

```
def Cube_form_I : Cube := I
def Cube_form_unit : Cube := 1
def Cube_form_prod (I J: Cube) : Cube := I × J
```

Definition 14 (Cube Introduction)**.** Cube terms are introduced:

$$\frac{\Xi \text{ cube ctx}}{\Xi \vdash * : 1} \quad \frac{\Xi \vdash s : I \quad \Xi \vdash t : J}{\Xi \vdash \langle s, t \rangle : I \times J}$$

```
def Cube_intro_unit : Cube_form_unit := *
def Cube_intro_pair (I J: Cube) (s: I) (t: J)
  : Cube_form_prod I J := <s, t>
```

Definition 15 (Cube Elimination)**.** Cube projections extract components:

$$\frac{\Xi \vdash t : I \times J}{\Xi \vdash \pi_1(t) : I} \quad \frac{\Xi \vdash t : I \times J}{\Xi \vdash \pi_2(t) : J}$$

```
def Cube_elim_fst (I J: Cube) (t: Cube_form_prod I J) : I := π₁ t
def Cube_elim_snd (I J: Cube) (t: Cube_form_prod I J) : J := π₂ t
```

1.4.3 Shape Topes

Definition 16 (Tope Formation)**.** Tope types encode logical constraints:

$$\frac{\Xi \text{ cube ctx}}{\Xi \vdash \top : \text{Tope}} \quad \frac{\Xi \text{ cube ctx}}{\Xi \vdash \bot : \text{Tope}} \quad \frac{\Xi \vdash s, t : I}{\Xi \vdash s \equiv t : \text{Tope}}$$

$$\frac{\Xi \vdash \phi : \text{Tope} \quad \Xi \vdash \psi : \text{Tope}}{\Xi \vdash \phi \wedge \psi : \text{Tope}} \quad \frac{\Xi \vdash \phi : \text{Tope} \quad \Xi \vdash \psi : \text{Tope}}{\Xi \vdash \phi \vee \psi : \text{Tope}}$$

```
def Tope_form_true : Tope := ⊤
def Tope_form_false : Tope := ⊥
def Tope_form_eq (I: Cube) (s t: I) : Tope := s ≡ t
def Tope_form_and (φ ψ: Tope) : Tope := φ ∧ ψ
def Tope_form_or (φ ψ: Tope) : Tope := φ ∨ ψ
```

Definition 17 (Tope Introduction)**.** Tope entailments are introduced:

$$\frac{\Xi \vdash \phi : \text{Tope}}{\Xi \vdash \phi \Rightarrow \top} \quad \frac{\Xi \vdash s \equiv t : \text{Tope}}{\Xi \vdash s \equiv t \Rightarrow t \equiv s}$$

$$\frac{\Xi \vdash s \equiv t \quad \Xi \vdash t \equiv u}{\Xi \vdash s \equiv u} \quad \frac{\Xi \vdash \phi : \text{Tope} \quad \Xi \vdash \psi : \text{Tope} \quad \Xi \vdash \phi \Rightarrow \psi \quad \Xi \vdash \psi \Rightarrow \chi}{\Xi \vdash \phi \Rightarrow \chi}$$

```
def Tope_intro_true (φ: Tope) : φ ⇒ ⊤ := λ _, ⊤
def Tope_intro_sym (I: Cube) (s t: I) : Tope_form_eq I s t ⇒
Tope_form_eq I t s
 := λ p, sym p
def Tope_intro_trans (I: Cube) (s t u: I)
  : Tope_form_eq I s t ⇒ Tope_form_eq I t u ⇒ Tope_form_eq I s u
 := λ p q, trans p q
```

Definition 18 (Tope Disjunction Elimination)**.** Tope disjunction is eliminated via a pushout-like rule:

$$\frac{\Xi \vdash \phi \vee \psi : \text{Tope} \quad \Xi, \phi \vdash \chi : \text{Tope} \quad \Xi, \psi \vdash \chi : \text{Tope} \quad \Xi, \phi \wedge \psi \vdash \chi : \text{Tope}}{\Xi \vdash \chi : \text{Tope}}$$

$$\begin{array}{ccc} \phi \wedge \psi & \longrightarrow & \psi \\ \downarrow & \chi & \downarrow \\ \phi & \longrightarrow & \phi \vee \psi \end{array}$$

```
def Tope_elim_or (φ ψ χ: Tope)
    (c1: φ ⇒ χ) (c2: ψ ⇒ χ) (c3: φ ∧ ψ ⇒ χ)
  : φ ∨ ψ ⇒ χ := λ _, χ
```

Theorem 11 (Tope Computation)**.** Disjunction elimination reduces on cases:

$$\frac{\Xi \vdash \phi : \text{Tope} \quad \Xi, \phi \vdash \chi : \text{Tope}}{\Xi \vdash \chi[\phi \vee \psi/\phi] \equiv \chi : \text{Tope}}$$

```
def Tope_comp_or (φ ψ χ: Tope) (c: φ ⇒ χ)
  : Ξ Tope (Tope_elim_or φ ψ χ c c c) χ := refl Tope χ
```

1.4.4 Extension Types

Definition 19 (Extension Type Formation)**.** Extension types generalize dependent functions:

$$\frac{\Xi \vdash \phi : \mathrm{Tope} \quad \Xi \vdash \psi : \mathrm{Tope} \quad \Xi \vdash i : \phi \hookrightarrow \psi \quad \Gamma \vdash C : \psi \to \mathcal{U} \quad \Gamma \vdash d : \prod_{x:\phi} C(i(x))}{\Gamma \vdash \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle : \mathcal{U}}$$

```
def Ext_form (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ
-> U) (d: (x: φ) -> C (i x))
  : U := <Π (y: ψ), C y | i>
```

Definition 20 (Extension Type Introduction)**.** Elements are functions satisfying boundary conditions:

$$\frac{\begin{array}{c}\Xi \vdash \phi : \mathrm{Tope} \quad \Xi \vdash \psi : \mathrm{Tope} \quad \Xi \vdash i : \phi \hookrightarrow \psi \\ \Gamma \vdash C : \psi \to \mathcal{U} \quad \Gamma \vdash f : \prod_{y:\psi} C(y) \quad \Gamma \vdash p : \prod_{x:\phi} f(i(x)) \equiv d(x)\end{array}}{\Gamma \vdash \mathrm{ext}(f, p) : \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle}$$

```
def Ext_intro (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ
-> U) (d: (x: φ) -> C (i x))
    (f: (y: ψ) -> C y) (p: (x: φ) -> Ξ (C (i x)) (f (i x)) (d x))
  : Ext_form φ ψ i C d := ext f p
```

Definition 21 (Extension Type Elimination)**.** Extension types are applied or restricted:

$$\frac{\Gamma \vdash e : \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle \quad \Gamma \vdash y : \psi}{\Gamma \vdash \mathrm{app}(e, y) : C(y)} \qquad \frac{\Gamma \vdash e : \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle \quad \Gamma \vdash x : \phi}{\Gamma \vdash \mathrm{restr}(e, x) : C(i(x))}$$

$$\begin{array}{ccc} \phi & \xrightarrow{i} & \psi \\ \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle C} \\ \mathcal{U} & \xrightarrow{\mathrm{ext}} & \mathcal{U} \end{array}$$

```
def Ext_elim_app (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ -> U)
    (d: (x: φ) -> C (i x)) (e: Ext_form φ ψ i C d) (y: ψ)
  : C y := app e y

def Ext_elim_restr (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ -> U)
    (d: (x: φ) -> C (i x)) (e: Ext_form φ ψ i C d) (x: φ)
  : C (i x) := restr e x
```

Theorem 12 (Extension Type Computation)**.** Application and restriction reduce appropriately:

$$\frac{\Gamma \vdash f : \prod_{y:\psi} C(y) \quad \Gamma \vdash p : \prod_{x:\phi} f(i(x)) \equiv d(x) \quad \Gamma \vdash y : \psi}{\Gamma \vdash \mathrm{app}(\mathrm{ext}(f, p), y) \equiv f(y) : C(y)}$$

$$\frac{\Gamma \vdash f : \prod_{y:\psi} C(y) \quad \Gamma \vdash p : \prod_{x:\phi} f(i(x)) \equiv d(x) \quad \Gamma \vdash x : \phi}{\Gamma \vdash \mathrm{restr}(\mathrm{ext}(f, p), x) \equiv d(x) : C(i(x))}$$

```
def Ext_comp_app (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ -> U)
    (d: (x: φ) -> C (i x)) (f: (y: ψ) -> C y)
    (p: (x: φ) -> Ξ (C (i x)) (f (i x)) (d x)) (y: ψ)
  : Ξ (C y) (Ext_elim_app φ ψ i C d (Ext_intro φ ψ
i C d f p) y) (f y)
 := refl (C y) (f y)

def Ext_comp_restr (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ -> U)
    (d: (x: φ) -> C (i x)) (f: (y: ψ) -> C y)
    (p: (x: φ) -> Ξ (C (i x)) (f (i x)) (d x)) (x: φ)
  : Ξ (C (i x)) (Ext_elim_restr φ ψ i C d (Ext_intro φ ψ
i C d f p) x) (d x)
 := p x
```

Theorem 13 (Extension Type Uniqueness)**.** Extension types are uniquely determined:

$$\frac{\Gamma \vdash e : \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle}{\Gamma \vdash e \equiv \mathrm{ext}(\lambda y.\mathrm{app}(e, y), \lambda x.\mathrm{restr}(e, x)) : \left\langle \prod_{y:\psi} C(y) \mid i \right\rangle}$$

```
def Ext_uniq (φ ψ: Tope) (i: φ ↪ ψ) (C: ψ
-> U) (d: (x: φ) -> C (i x))
    (e: Ext_form φ ψ i C d)
  : Ξ (Ext_form φ ψ i C d) e
      (Ext_intro φ ψ i C d (λ y, Ext_elim_app φ ψ i C d e y)
                           (λ x, Ext_elim_restr φ ψ i C d e x))
 := refl (Ext_form φ ψ i C d) e
```

1.4.5 Universe Types

Definition 22 (Universe Formation)**.** The universe type is formed as:

$$\frac{\Gamma \ \mathrm{ctx}}{\Gamma \vdash \mathcal{U} : \mathcal{U}_{\mathrm{succ}}}$$

```
def Univ_form : U₁ := U
```

Definition 23 (Universe Introduction)**.** Types are elements of the universe:

$$\frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash A : \mathcal{U}}$$

```
def Univ_intro (A: U) : Univ_form := A
```

Definition 24 (Universe Elimination)**.** Types in the universe are used in type formation:

$$\frac{\Gamma \vdash A : \mathcal{U} \quad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \Pi_{x:A} B : \mathcal{U}}$$

```
def Univ_elim (A: Univ_form) (B: A -> U) : Univ_form := Π
(x: A), B x
```

Theorem 14 (Universe Computation)**.** Universe elimination aligns with type formation:

$$\frac{\Gamma \vdash A : \mathcal{U} \quad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \mathrm{Univ\_elim}(A, B) \equiv \Pi_{x:A} B : \mathcal{U}}$$

```
def Univ_comp (A: U) (B: A -> U)
  : Ξ U (Univ_elim A B) (Π (x: A), B x)
 := refl U (Π (x: A), B x)
```

1.5 Simplicial Type Theory

This document formalizes the inference rules for Martin-Löf Type Theory (MLTT) and its extension to Simplicial Type Theory (STT) as presented in "The Yoneda embedding in simplicial type theory" (2025) by Gratzer, Weinberger, and Buchholtz.

1.5.1 Judgments

The type system uses:

- $\vdash \Gamma$: Context Γ is well-formed.
- $\Gamma \vdash \delta : \Delta$: Substitution δ maps Γ to Δ.
- $\Gamma \vdash A$ type: Type A is well-formed in Γ.
- $\Gamma \vdash a : A$: Term a has type A in Γ.

1.5.2 Context Formation with Extension

- Modal context extension: $\vdash \Gamma \implies \vdash \Gamma, \{\mu\}$
- Variable annotation: $\vdash \Gamma, \Gamma, \{\mu\} \vdash A$ type $\implies \vdash \Gamma, x :_\mu A$
- Variable rule: $\mu \leq \mathrm{mods}(\Gamma_1) \implies \Gamma_0, x :_\mu A, \Gamma_1 \vdash x : A$

where $\mathrm{mods}(\Gamma_1) = \nu_0 \circ \nu_1 \circ \cdots$ is the composite of modal restrictions $\{\nu_i\}$ in Γ_1 (or id if none).

1.5.3 Dependent Function Types (Π-Types)

Definition (Inference Rules):

- Formation: $\Gamma \vdash A$ type, $\Gamma, x : A \vdash B$ type $\implies \Gamma \vdash \prod_{x:A} B$ type
- Introduction: $\Gamma, x : A \vdash b : B \implies \Gamma \vdash \lambda x.b : \prod_{x:A} B$
- Elimination: $\Gamma \vdash f : \prod_{x:A} B, \Gamma \vdash a : A \implies \Gamma \vdash f(a) : B[a/x]$
- Computation: $\Gamma, x : A \vdash b : B, \Gamma \vdash a : A \implies \Gamma \vdash (\lambda x.b)(a) \equiv b[a/x] : B[a/x]$
- Uniqueness: $\Gamma \vdash f : \prod_{x:A} B \implies \Gamma \vdash \lambda x.f(x) \equiv f : \prod_{x:A} B$

Theorem (Type Safety): For any Γ, if $\Gamma \vdash \prod_{x:A} B$ type and $\Gamma \vdash f : \prod_{x:A} B$, then for any $\Gamma \vdash a : A$, there exists a unique $b : B[a/x]$ such that $\Gamma \vdash f(a) \equiv b : B[a/x]$.
Proof Sketch: Formation ensures A, B are well-typed. Introduction constructs f. Elimination applies f to a. Computation reduces $f(a)$. Uniqueness follows from the η-rule.

1.5.4 Dependent Pair Types (Σ-Types)

Definition (Inference Rules):

- Formation: $\Gamma \vdash A \text{ type}, \Gamma, x : A \vdash B \text{ type} \implies \Gamma \vdash \sum_{x:A} B \text{ type}$
- Introduction: $\Gamma \vdash a : A, \Gamma \vdash b : B[a/x] \implies \Gamma \vdash (a, b) : \sum_{x:A} B$
- Elimination: $\Gamma, z : \sum_{x:A} B \vdash C \text{ type}, \Gamma, x : A, y : B \vdash c : C[(x, y)/z], \Gamma \vdash p : \sum_{x:A} B \implies \Gamma \vdash \text{let } (x, y) \leftarrow p \text{ in } c : C[p/z]$
- Computation: $\Gamma, z : \sum_{x:A} B \vdash C \text{ type}, \Gamma, x : A, y : B \vdash c : C[(x, y)/z], \Gamma \vdash a : A, \Gamma \vdash b : B[a/x] \implies \Gamma \vdash \text{let } (x, y) \leftarrow (a, b) \text{ in } c \equiv c[a/x, b/y] : C[(a, b)/z]$
- Uniqueness: $\Gamma \vdash p : \sum_{x:A} B \implies \Gamma \vdash \text{let } (x, y) \leftarrow p \text{ in } (x, y) \equiv p : \sum_{x:A} B$

Theorem (Fibration Property): $\Gamma \vdash \sum_{x:A} B$ type is a fibration in the locally cartesian closed category of contexts.
Proof Sketch: Formation defines the fibration. Introduction constructs sections. Elimination performs dependent elimination. Computation ensures coherence. Uniqueness reflects the universal property.

1.5.5 Universes

Definition (Inference Rules):

- Formation: $\text{true} \implies \Gamma \vdash \mathcal{U}_i \text{ type}$
- Introduction: $\Gamma \vdash A \text{ type}, \text{level}(A) \leq i \implies \Gamma \vdash A : \mathcal{U}_i$
- Elimination: $\Gamma \vdash A : \mathcal{U}_i \implies \Gamma \vdash \text{El}(A) \text{ type}$
- Computation: $\Gamma \vdash A : \mathcal{U}_i, \Gamma \vdash A \text{ type} \implies \Gamma \vdash \text{El}(A) \equiv A \text{ type}$
- Uniqueness: $\Gamma \vdash A : \mathcal{U}_i \implies \Gamma \vdash A \equiv \text{El}(A) : \mathcal{U}_i$

Theorem (Consistency): The hierarchy $\mathcal{U}_i$ ensures MLTT consistency by stratifying types.
Proof Sketch: Formation introduces the hierarchy. Introduction embeds types. Elimination decodes them. Computation ensures idempotence. Uniqueness guarantees coherence.

1.5.6 Interval Type (𝕀)

Definition[Inference Rules]

- Formation: $\text{true} \implies \Gamma \vdash \mathbb{I} \text{ type}$
- Introduction: $\text{true} \implies \Gamma \vdash 0 : \mathbb{I}, \text{true} \implies \Gamma \vdash 1 : \mathbb{I}, \Gamma \vdash i : \mathbb{I}, \Gamma \vdash j : \mathbb{I} \implies \Gamma \vdash i \wedge j : \mathbb{I}, \Gamma \vdash i : \mathbb{I}, \Gamma \vdash j : \mathbb{I} \implies \Gamma \vdash i \vee j : \mathbb{I}$

- Elimination: $\Gamma, x : \mathbb{I} \vdash B \text{ type}, \Gamma \vdash b_0 : B[0/x], \Gamma \vdash b_1 : B[1/x], \Gamma, x, y : \mathbb{I} \vdash b_\wedge : B[x \wedge y/x], \Gamma, x, y : \mathbb{I} \vdash b_\vee : B[x \vee y/x], \Gamma \vdash i : \mathbb{I} \implies \Gamma \vdash \mathrm{rec}_{\mathbb{I}}(B, b_0, b_1, b_\wedge, b_\vee, i) : B[i/x]$
- Computation: (as above) $\implies \Gamma \vdash \mathrm{rec}_{\mathbb{I}}(B, b_0, b_1, b_\wedge, b_\vee, 0) \equiv b_0 : B[0/x]$, $\ldots$, $\Gamma \vdash \mathrm{rec}_{\mathbb{I}}(B, b_0, b_1, b_\wedge, b_\vee, i \vee j) \equiv b_\vee[i/x, j/y] : B[i \vee y/x]$
- Uniqueness: $\Gamma \vdash f : \prod_{x:\mathbb{I}} B, \Gamma \vdash f(0) \equiv b_0 : B[0/x], \ldots, \Gamma \vdash f(x \vee y) \equiv b_\vee : B[x \vee y/x] \implies \Gamma \vdash \mathrm{rec}_{\mathbb{I}}(B, b_0, b_1, b_\wedge, b_\vee, i)$ unique up to $\equiv$

Theorem[Bounded Lattice] $\mathbb{I}$ forms a bounded distributive lattice, satisfying $\prod_{i,j:\mathbb{I}}(i \leq j) \vee (j \leq i)$ (Axiom A).
Proof Sketch: Introduction defines lattice operations. Elimination and computation ensure well-definedness. Uniqueness respects the lattice structure.

1.5.7 Modal Types ($\langle \mu \mid A \rangle$)

Definition[Inference Rules]

- Formation: $\Gamma, \{\mu\} \vdash A \text{ type} \implies \Gamma \vdash \langle \mu \mid A \rangle \text{ type}$
- Introduction: $\Gamma, \{\mu\} \vdash a : A \implies \Gamma \vdash \mathrm{mod}_\mu(a) : \langle \mu \mid A \rangle$
- Elimination: $\Gamma, x :_\nu \langle \mu \mid A \rangle \vdash B \text{ type}, \Gamma, y :_{\nu \circ \mu} A \vdash b : B[\mathrm{mod}_\mu(y)/x], \Gamma, \{\nu\} \vdash a : \langle \mu \mid A \rangle \implies \Gamma \vdash \mathrm{let}\ \mathrm{mod}_\mu(y) \leftarrow a\ \mathrm{in}\ b : B[a/x]$
- Computation: $\Gamma, x :_\nu \langle \mu \mid A \rangle \vdash B \text{ type}, \Gamma, y :_{\nu \circ \mu} A \vdash b : B[\mathrm{mod}_\mu(y)/x], \Gamma, \{\nu \circ \mu\} \vdash a : A \implies \Gamma \vdash \mathrm{let}\ \mathrm{mod}_\mu(y) \leftarrow \mathrm{mod}_\mu(a)\ \mathrm{in}\ b \equiv b[a/y] : B[\mathrm{mod}_\mu(a)/x]$
- Uniqueness: $\Gamma \vdash f : \langle \mu \mid A \rangle \to B, \Gamma, y :_{\nu \circ \mu} A \vdash f(\mathrm{mod}_\mu(y)) \equiv b : B[\mathrm{mod}_\mu(y)/x], \Gamma, \{\nu\} \vdash a : \langle \mu \mid A \rangle \implies \Gamma \vdash \mathrm{let}\ \mathrm{mod}_\mu(y) \leftarrow a\ \mathrm{in}\ b$ unique up to $\equiv$

Theorem[Modal Equivalence] For μ, $\mathrm{mod}_\mu : A \to \langle \mu \mid A \rangle$ is an equivalence if $\mathrm{mod}_\mu(a) = \mathrm{mod}_\mu(b) \to \langle \mu \mid a = b \rangle$ is an equivalence (Axiom B).
Proof Sketch: Formation and introduction define the modal type. Elimination and computation ensure injectivity. Axiom B guarantees surjectivity.

1.5.8 Modal Π-Types

Definition[Inference Rules]

- Formation: $\Gamma \vdash A \text{ type}, \Gamma, x :_\mu A \vdash B \text{ type} \implies \Gamma \vdash \prod_{x:_\mu A} B \text{ type}$
- Introduction: $\Gamma, x :_\mu A \vdash b : B \implies \Gamma \vdash \lambda x.b : \prod_{x:_\mu A} B$
- Elimination: $\Gamma \vdash f : \prod_{x:_\mu A} B, \Gamma, \{\mu\} \vdash a : A \implies \Gamma \vdash f(a) : B[a/x]$
- Computation: $\Gamma, x :_\mu A \vdash b : B, \Gamma, \{\mu\} \vdash a : A \implies \Gamma \vdash (\lambda x.b)(a) \equiv b[a/x] : B[a/x]$

- Uniqueness: $\Gamma \vdash \mathsf{f} : \prod_{\mathsf{x}:_\mu \mathsf{A}} \mathsf{B} \implies \Gamma \vdash \lambda \mathsf{x}.\mathsf{f}(\mathsf{x}) \equiv \mathsf{f} : \prod_{\mathsf{x}:_\mu \mathsf{A}} \mathsf{B}$

Theorem[Pointwise Invertibility] For a category C, if $\mathsf{f} : \prod_{\mathsf{x}:_\flat \mathsf{C}} \langle \flat \mid \hom_{\mathsf{C}}(\mathsf{x}, \mathsf{y}) \rangle$ is pointwise invertible, then f is globally invertible (Example 2.22).
Proof Sketch: Modal Π-types ensure f respects $\flat$. Elimination applies f. Computation preserves equalities. The $\flat$-modality extracts isomorphisms.

1.5.9 Precategory and Category Types

Definition[Inference Rules]

- Formation (Precategory): $\Gamma \vdash \mathsf{C}\ \text{type}, \Gamma \vdash \text{isSegal}(\mathsf{C}) : \prod_{\mathsf{x},\mathsf{y},\mathsf{z}:\mathsf{C}} (\mathbb{I} \to \hom_{\mathsf{C}}(\mathsf{x}, \mathsf{y})) \times (\mathbb{I} \to \hom_{\mathsf{C}}(\mathsf{y}, \mathsf{z})) \to \hom_{\mathsf{C}}(\mathsf{x}, \mathsf{z}) \implies \Gamma \vdash \mathsf{C}\ \text{precategory}$
- Formation (Category): $\Gamma \vdash \mathsf{C}\ \text{precategory}, \Gamma \vdash \text{isRezk}(\mathsf{C}) : \prod_{\mathsf{x},\mathsf{y}:\mathsf{C}} \text{isEquiv}(\text{iso}_{\mathsf{x},\mathsf{y}} \to (\mathsf{x} = \mathsf{y})) \implies \Gamma \vdash \mathsf{C}\ \text{category}$

where $\hom_{\mathsf{C}}(\mathsf{x}, \mathsf{y}) = \langle \sharp \mid \mathsf{C} \rangle$, $\text{iso}_{\mathsf{x},\mathsf{y}} = \langle \flat \mid \hom_{\mathsf{C}}(\mathsf{x}, \mathsf{y}) \rangle$.

Theorem[Yoneda Embedding] For a category C, there exists a fully faithful $\mathbf{y} : \mathsf{C} \to \widehat{\mathsf{C}}$, where $\widehat{\mathsf{C}}$ is the precategory of presheaves $\langle \sharp \mid \mathsf{C} \rangle$, defined by $\mathbf{y}(\mathsf{x})(\mathsf{y}) = \hom_{\mathsf{C}}(\mathsf{y}, \mathsf{x})$.
Proof Sketch: Precategory formation defines $\widehat{\mathsf{C}}$. Yoneda uses modal types and $\sharp$. Rezk ensures $\hom_{\widehat{\mathsf{C}}}(\mathbf{y}(\mathsf{x}), \mathbf{y}(\mathsf{y})) \simeq \hom_{\mathsf{C}}(\mathsf{x}, \mathsf{y})$.

1.5.10 Key Theorems

- **Yoneda Lemma**: For a category C, $\hom_{\widehat{\mathsf{C}}}(\mathbf{y}(\mathsf{x}), \mathsf{F}) \to \mathsf{F}(\mathsf{x})$ is an equivalence for all $\mathsf{x} : \mathsf{C}$, $\mathsf{F} : \langle \sharp \mid \mathsf{C} \rangle$.
  Proof Sketch: Uses tw modality and Axiom G for twisted arrow categories.
- **Free Cocompletion**: $\widehat{\mathsf{C}}$ is the free cocompletion of C, i.e., $\text{Fun}(\widehat{\mathsf{C}}, \mathsf{D}) \simeq \text{Fun}(\mathsf{C}, \mathsf{D})$ for cocomplete D.
  Proof Sketch: Modal Π-types and $\sharp$ define functor categories. Yoneda ensures universality.

Література

[1] Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book/`, 2013.

[2] Kapulkin, C., LeFanu Lumsdaine, P., Voevodsky, V. *The simplicial model of univalent foundations*. arXiv:1211.2851, 2012.

[3] Shulman, M. *The univalence axiom for elegant Reedy presheaves*. Homology, Homotopy, and Applications, 17(2):81–106, 2015. arXiv:1307.6248.

[4] Joyal, A., Tierney, M. *Quasi-categories vs Segal spaces*. arXiv:math/0607820, 2006.

[5] Johnstone, P. T. *On a topological topos*. Proc. London Math. Soc., 38:237–271, 1979.

[6] Mac Lane, S., Moerdijk, I. *Sheaves in geometry and logic*. Springer, 1994.

[7] Rezk, C. *A model for the homotopy theory of homotopy theory*. Trans. Amer. Math. Soc., 353(3):973–1007, 2001. arXiv:math.AT/9811037.

[8] Emily Riehl, Michael Shulman. *A type theory for synthetic -categories*. Arxiv. 2017

[9] Daniel Gratzer, Jonathan Weinberger, Ulrik Buchholtz. *The Yoneda embedding in simplicial type theory*. Arxiv. 2025