A synthetic theory of $\infty$-categories in homotopy type theory

joint with Michael Shulman

Workshop on Higher Category Approach to Certifiably Correct Quantum Information Processing Systems
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.

The schematic notion of $\infty$-category is made precise by several models: quasi-categories, Rezk spaces, Segal categories, 1-complicial sets, …
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.

The schematic notion of $\infty$-category is made precise by several models: quasi-categories, Rezk spaces, Segal categories, 1-complicial sets, …
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.
The idea of an $\infty$-category

An $\infty$-category, a nickname for an $(\infty, 1)$-category, has:

- objects
- 1-arrows between these objects
- with composites of these 1-arrows witnessed by invertible 2-arrows
- with composition associative up to invertible 3-arrows (and unital)
- with these witnesses coherent up to invertible arrows all the way up

The composition operation and associativity and unit axioms in a 1-category become higher data in an $(\infty, 1)$-category.

The schematic notion of $\infty$-category is made precise by several models: quasi-categories, Rezk spaces, Segal categories, 1-complicial sets, …
The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:
• work analytically to give categorical definitions and prove theorems using the combinatorics of one model (e.g., Joyal, Lurie, Gepner-Haugseng, Cisinski in $\text{qC}$; Kazhdan-Varshavsky, Rasekh in $\text{R}$; Simpson in $\text{Segal}$)
• work synthetically to give categorical definitions and prove theorems in all four models $\text{qC}$, $\text{R}$, $\text{Segal}$, $\text{1-Comp}$ at once (R-Verity: an $\infty$-cosmos axiomatizes the common features of the categories $\text{qC}$, $\text{R}$, $\text{Segal}$, $\text{1-Comp}$ of $\infty$-categories)
• work synthetically in a simplicial type theory augmenting homotopy type theory to prove theorems in $\text{R}$ (R-Shulman: an $\infty$-category is a type with unique binary composites in which isomorphism is equivalent to identity)
The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work analytically to give categorical definitions and prove theorems using the combinatorics of one model
  
  (eg., Joyal, Lurie, Gepner-Haugsseng, Cisinski in $q\text{Cat}$; Kazhdan-Varshavsky, Rasekh in $\text{Rezk}$; Simpson in $\text{Segal}$)
The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work **analytically** to give categorical definitions and prove theorems using the combinatorics of one model
  
  (eg., Joyal, Lurie, Gepner-Haugse, Cisinski in $q\text{Cat}$; Kazhdan-Varshavsky, Rasekh in $\text{Rezk}$; Simpson in $\text{Segal}$)

- work **synthetically** to give categorical definitions and prove theorems in all four models $q\text{Cat}$, $\text{Rezk}$, $\text{Segal}$, $1\text{-Comp}$ at once
  
  (R-Verity: an $\infty$-cosmos axiomatizes the common features of the categories $q\text{Cat}$, $\text{Rezk}$, $\text{Segal}$, $1\text{-Comp}$ of $\infty$-categories)
The analytic vs synthetic theory of $\infty$-categories

Q: How might you develop the category theory of $\infty$-categories?

Strategies:

- work **analytically** to give categorical definitions and prove theorems using the combinatorics of one model
  
  (eg., Joyal, Lurie, Gepner-Haugseng, Cisinski in $q\text{Cat}$; Kazhdan-Varshavsky, Rasekh in $\text{Rezk}$; Simpson in $\text{Segal}$)

- work **synthetically** to give categorical definitions and prove theorems in all four models $q\text{Cat}$, $\text{Rezk}$, $\text{Segal}$, $1\text{-Comp}$ at once
  
  (R-Verity: an $\infty$-cosmos axiomatizes the common features of the categories $q\text{Cat}$, $\text{Rezk}$, $\text{Segal}$, $1\text{-Comp}$ of $\infty$-categories)

- work **synthetically** in a simplicial type theory augmenting homotopy type theory to prove theorems in $\text{Rezk}$
  
  (R-Shulman: an $\infty$-category is a type with unique binary composites in which isomorphism is equivalent to identity)
Plan

1. Homotopy type theory

2. A directed type theory for $\infty$-categories

3. The synthetic theory of $\infty$-categories
Homotopy type theory
Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$

including in particular identity types $x, y : A \vdash x = A y$.

Type constructors build new types and terms from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$
- dependent sums $\sum x : A B(x)$, dependent products $\prod x : A B(x)$.

Each type constructor comes with rules:

1. formation: a way to construct new types
2. introduction: ways to construct terms of these types
3. elimination: ways to use them to construct other terms
4. computation: what happens when we follow (ii) by (iii)
Types, terms, and type constructors

Homotopy type theory has:

- types $A, B$
- terms $x : A, y : B$
Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$
Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$. 

Type constructors build new types and terms from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$,
- dependent sums $\sum x : A B(x)$, dependent products $\prod x : A B(x)$.

Each type constructor comes with rules:

(i) formation: a way to construct new types
(ii) introduction: ways to construct terms of these types
(iii) elimination: ways to use them to construct other terms
(iv) computation: what happens when we follow (ii) by (iii)
Types, terms, and type constructors

Homotopy type theory has:

- types \( A, B \)
- terms \( x : A, y : B \)
- dependent types \( x : A \vdash B(x), x, y : A \vdash B(x, y) \) including in particular identity types \( x, y : A \vdash x \equiv_A y \).

Type constructors build new types and terms from given ones:
Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$, 
Types, terms, and type constructors

Homotopy type theory has:

- types $A$, $B$
- terms $x : A$, $y : B$
- dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

- products $A \times B$, coproducts $A + B$, function types $A \to B$,
- dependent sums $\sum_{x:A} B(x)$, dependent products $\prod_{x:A} B(x)$. 
Types, terms, and type constructors

Homotopy type theory has:

• types $A, B$
• terms $x : A$, $y : B$
• dependent types $x : A \vdash B(x)$, $x, y : A \vdash B(x, y)$ including in particular identity types $x, y : A \vdash x =_A y$.

Type constructors build new types and terms from given ones:

• products $A \times B$, coproducts $A + B$, function types $A \to B$,
• dependent sums $\sum_{x : A} B(x)$, dependent products $\prod_{x : A} B(x)$.

Each type constructor comes with rules:

(i) formation: a way to construct new types
(ii) introduction: ways to construct terms of these types
(iii) elimination: ways to use them to construct other terms
(iv) computation: what happens when we follow (ii) by (iii)
The extended Curry-Howard correspondence

<table>
<thead>
<tr>
<th>type theory</th>
<th>set theory</th>
<th>logic</th>
<th>homotopy theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>$x : A$</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
</tbody>
</table>
## The extended Curry-Howard correspondence

<table>
<thead>
<tr>
<th>type theory</th>
<th>set theory</th>
<th>logic</th>
<th>homotopy theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>( x : A )</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
<tr>
<td>( \emptyset, 1 )</td>
<td>( \emptyset, {\emptyset} )</td>
<td>( \bot, \top )</td>
<td>( \emptyset, * )</td>
</tr>
<tr>
<td>( A \times B )</td>
<td>set of pairs</td>
<td>( A ) and ( B )</td>
<td>product space</td>
</tr>
<tr>
<td>( A + B )</td>
<td>disjoint union</td>
<td>( A ) or ( B )</td>
<td>coproduct</td>
</tr>
<tr>
<td>( A \rightarrow B )</td>
<td>set of functions</td>
<td>( A ) implies ( B )</td>
<td>function space</td>
</tr>
</tbody>
</table>
The extended Curry-Howard correspondence

<table>
<thead>
<tr>
<th>type theory</th>
<th>set theory</th>
<th>logic</th>
<th>homotopy theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>$x : A$</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
<tr>
<td>$\emptyset, 1$</td>
<td>$\emptyset, {\emptyset}$</td>
<td>$\bot, \top$</td>
<td>$\emptyset, \ast$</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>set of pairs</td>
<td>$A$ and $B$</td>
<td>product space</td>
</tr>
<tr>
<td>$A + B$</td>
<td>disjoint union</td>
<td>$A$ or $B$</td>
<td>coproduct</td>
</tr>
<tr>
<td>$A \rightarrow B$</td>
<td>set of functions</td>
<td>$A$ implies $B$</td>
<td>function space</td>
</tr>
<tr>
<td>$x : A \vdash B(x)$</td>
<td>family of sets</td>
<td>predicate</td>
<td>fibration</td>
</tr>
</tbody>
</table>
## The extended Curry-Howard correspondence

<table>
<thead>
<tr>
<th>type theory</th>
<th>set theory</th>
<th>logic</th>
<th>homotopy theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>( x : A )</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
<tr>
<td>( \emptyset, 1 )</td>
<td>( \emptyset, {\emptyset} )</td>
<td>( \perp, \top )</td>
<td>( \emptyset, \ast )</td>
</tr>
<tr>
<td>( A \times B )</td>
<td>set of pairs</td>
<td>( A \text{ and } B )</td>
<td>product space</td>
</tr>
<tr>
<td>( A + B )</td>
<td>disjoint union</td>
<td>( A \text{ or } B )</td>
<td>coproduct</td>
</tr>
<tr>
<td>( A \rightarrow B )</td>
<td>set of functions</td>
<td>( A \text{ implies } B )</td>
<td>function space</td>
</tr>
<tr>
<td>( x : A \vdash B(x) )</td>
<td>family of sets</td>
<td>predicate</td>
<td>fibration</td>
</tr>
<tr>
<td>( x : A \vdash b : B(x) )</td>
<td>fam. of elements</td>
<td>conditional proof</td>
<td>section</td>
</tr>
<tr>
<td>type theory</td>
<td>set theory</td>
<td>logic</td>
<td>homotopy theory</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>$A$</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>$x : A$</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
<tr>
<td>$\emptyset, 1$</td>
<td>$\emptyset, {\emptyset}$</td>
<td>$\bot, \top$</td>
<td>$\emptyset, \ast$</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>set of pairs</td>
<td>$A \text{ and } B$</td>
<td>product space</td>
</tr>
<tr>
<td>$A + B$</td>
<td>disjoint union</td>
<td>$A \text{ or } B$</td>
<td>coproduct</td>
</tr>
<tr>
<td>$A \rightarrow B$</td>
<td>set of functions</td>
<td>$A \text{ implies } B$</td>
<td>function space</td>
</tr>
<tr>
<td>$x : A \vdash B(x)$</td>
<td>family of sets</td>
<td>conditional proof</td>
<td>fibration</td>
</tr>
<tr>
<td>$x : A \vdash b : B(x)$</td>
<td>fam. of elements</td>
<td>$\forall x. B(x)$</td>
<td>section</td>
</tr>
<tr>
<td>$\prod_{x:A} B(x)$</td>
<td>product</td>
<td>$\exists x. B(x)$</td>
<td>space of sections</td>
</tr>
<tr>
<td>$\sum_{x:A} B(x)$</td>
<td>disjoint sum</td>
<td></td>
<td>total space</td>
</tr>
</tbody>
</table>
### The extended Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Type theory</th>
<th>Set theory</th>
<th>Logic</th>
<th>Homotopy theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>set</td>
<td>proposition</td>
<td>space</td>
</tr>
<tr>
<td>$x : A$</td>
<td>element</td>
<td>proof</td>
<td>point</td>
</tr>
<tr>
<td>$\emptyset, 1$</td>
<td>$\emptyset, {\emptyset}$</td>
<td>$\bot, \top$</td>
<td>$\emptyset,*$</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>set of pairs</td>
<td>$A$ and $B$</td>
<td>product space</td>
</tr>
<tr>
<td>$A + B$</td>
<td>disjoint union</td>
<td>$A$ or $B$</td>
<td>coproduct</td>
</tr>
<tr>
<td>$A \rightarrow B$</td>
<td>set of functions</td>
<td>$A$ implies $B$</td>
<td>function space</td>
</tr>
<tr>
<td>$x : A \vdash B(x)$</td>
<td>family of sets</td>
<td>predicate</td>
<td>fibration</td>
</tr>
<tr>
<td>$x : A \vdash b : B(x)$</td>
<td>fam. of elements</td>
<td>conditional proof</td>
<td>section</td>
</tr>
<tr>
<td>$\prod_{x:A} B(x)$</td>
<td>product</td>
<td>$\forall x.B(x)$</td>
<td>space of sections</td>
</tr>
<tr>
<td>$\sum_{x:A} B(x)$</td>
<td>disjoint sum</td>
<td>$\exists x.B(x)$</td>
<td>total space</td>
</tr>
<tr>
<td>$p : x =_A y$</td>
<td>$x = y$</td>
<td>proof of equality</td>
<td>path from $x$ to $y$</td>
</tr>
<tr>
<td>$\sum_{x,y:A} x =_A y$</td>
<td>diagonal</td>
<td>equality relation</td>
<td>path space for $A$</td>
</tr>
</tbody>
</table>
Identity types and path induction

Formation rule for identity types

\[
\begin{array}{c}
x, y : A \\
x =_A y \text{ type}
\end{array}
\]
Identity types and path induction

<table>
<thead>
<tr>
<th>Formation and introduction rules for identity types</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x, y : A$</td>
</tr>
<tr>
<td>$\frac{}{x =_A y \text{ type}}$</td>
</tr>
<tr>
<td>$x : A$</td>
</tr>
<tr>
<td>$\frac{}{\text{refl}_x : x =_A x}$</td>
</tr>
</tbody>
</table>

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms $\text{refl}_x : x =_A x$ — is packaged in a proof technique called path induction.
Identity types and path induction

Formation and introduction rules for identity types

\[
\begin{align*}
\frac{x, y : A}{x =_A y \text{ type}} \\
\frac{x : A}{\text{refl}_x : x =_A x}
\end{align*}
\]

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms \(\text{refl}_x : x =_A x\) — is packaged in a proof technique called path induction.

Path induction. If \(B(x, y, p)\) is a type family dependent on \(x, y : A\) and \(p : x =_A y\), then to prove \(B(x, y, p)\) it suffices to assume \(y\) is \(x\) and \(p\) is \(\text{refl}_x\).
Identity types and path induction

Formation and introduction rules for identity types

\[
\begin{align*}
&\frac{x, y : A}{x =_A y \text{ type}} \\
&\frac{x : A}{\text{refl}_x : x =_A x}
\end{align*}
\]

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms \(\text{refl}_x : x =_A x\) — is packaged in a proof technique called path induction.

Path induction. If \(B(x, y, p)\) is a type family dependent on \(x, y : A\) and \(p : x =_A y\), then to prove \(B(x, y, p)\) it suffices to assume \(y\) is \(x\) and \(p\) is \(\text{refl}_x\). I.e., there is a function

\[
\text{path-ind} : \left( \prod_{x : A} B(x, x, \text{refl}_x) \right) \to \left( \prod_{x, y : A} \prod_{p : x =_A y} B(x, y, p) \right).
\]
Identity types and path induction

Formation and introduction rules for identity types

\[
\begin{align*}
\frac{x, y : A}{x =_A y \text{ type}} \quad &\quad \frac{x : A}{\text{refl}_x : x =_A x}
\end{align*}
\]

The elimination rule for the identity type — which says that the identity type family is freely generated by the terms \(\text{refl}_x : x =_A x\) — is packaged in a proof technique called path induction.

Path induction. If \(B(x, y, p)\) is a type family dependent on \(x, y : A\) and \(p : x =_A y\), then to prove \(B(x, y, p)\) it suffices to assume \(y\) is \(x\) and \(p\) is \(\text{refl}_x\). I.e., there is a function

\[
\text{path-ind} : \left( \prod_{x : A} B(x, x, \text{refl}_x) \right) \rightarrow \left( \prod_{x, y : A} \prod_{p : x =_A y} B(x, y, p) \right).
\]
The $\infty$-groupoid of paths

Theorem (Lumsdaine, Garner–van den Berg). The terms belonging to the iterated identity types of any type $A$ form an $\infty$-groupoid.
The $\infty$-groupoid of paths

Theorem (Lumsdaine, Garner–van den Berg). The terms belonging to the iterated identity types of any type $A$ form an $\infty$-groupoid.

The $\infty$-groupoid structure of $A$ has

- terms $x : A$ as objects
- paths $p : x =_A y$ as 1-morphisms
- paths of paths $h : p =_x =_A y q$ as 2-morphisms, \ldots

The required structures are proven from the path induction principle:

- constant paths (reflexivity) $\text{refl}_x : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $q \circ p : x = z$ and furthermore
  - concatenation is associative
  - the associators are coherent, \ldots
The $\infty$-groupoid of paths

Theorem (Lumsdaine, Garner–van den Berg). The terms belonging to the iterated identity types of any type $A$ form an $\infty$-groupoid.

The $\infty$-groupoid structure of $A$ has

- terms $x : A$ as objects
- paths $p : x =_{A} y$ as 1-morphisms
- paths of paths $h : p =_{x =_{A} y} q$ as 2-morphisms, ... 

The required structures are proven from the path induction principle:

- constant paths (reflexivity) $\text{refl}_{x} : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $q \ast p : x = z$
The $\infty$-groupoid of paths

Theorem (Lumsdaine, Garner–van den Berg). The terms belonging to the iterated identity types of any type $A$ form an $\infty$-groupoid. 

The $\infty$-groupoid structure of $A$ has

- terms $x : A$ as objects
- paths $p : x =_A y$ as 1-morphisms
- paths of paths $h : p =_{x =_A y} q$ as 2-morphisms, ...

The required structures are proven from the path induction principle:

- constant paths (reflexivity) $\text{refl}_x : x = x$
- reversal (symmetry) $p : x = y$ yields $p^{-1} : y = x$
- concatenation (transitivity) $p : x = y$ and $q : y = z$ yield $q \ast p : x = z$

and furthermore

- concatenation is associative
- the associators are coherent, ...
A directed type theory for ∞-categories
Shapes in the theory of the directed interval

Our types may depend on other types and also on shapes $\Phi \subset 2^n$, polytopes embedded in a directed cube, defined in a language $\top, \bot, \land, \lor, \equiv$ and $0, 1, \leq$ satisfying intuitionistic logic and strict interval axioms.
Shapes in the theory of the directed interval

Our types may depend on other types and also on shapes $\Phi \subset 2^n$, polytopes embedded in a directed cube, defined in a language

$$\top, \bot, \land, \lor, \equiv \quad \text{and} \quad 0, 1, \leq$$

satisfying intuitionistic logic and strict interval axioms.

$$\Delta^n := \{(t_1, \ldots, t_n) : 2^n \mid t_n \leq \cdots \leq t_1\}$$

e.g. $\Delta^1 := 2$ \quad $\Delta^2 := \left\{ \begin{array}{ll} (t,t) & (1,1) \\
(0,0) & (1,0) \\
(t,0) & (1,t) \end{array} \right.$

$$\partial \Delta^2 := \{(t_1, t_2) : 2^2 \mid (t_2 \leq t_1) \land ((0 \equiv t_2) \lor (t_2 \equiv t_1) \lor (t_1 \equiv 1))\}$$

$$\Lambda^2_1 := \{(t_1, t_2) : 2^2 \mid (t_2 \leq t_1) \land ((0 \equiv t_2) \lor (t_1 \equiv 1))\}$$

Because $\phi \land \psi$ implies $\phi$, there are shape inclusions $\Lambda^2_1 \subset \partial \Delta^2 \subset \Delta^2$. 
Extension types

Formation rule for extension types

\[ \Phi \subset \Psi \text{ shape} \quad A \text{ type} \quad a : \Phi \rightarrow A \]

\[ \langle \Phi \downarrow \Psi \quad a \quad A \rangle \text{ type} \]
Extension types

Formation rule for extension types

\[ \Phi \subset \Psi \text{ shape} \quad A \text{ type} \quad a : \Phi \to A \]

\[ \langle \Phi \vdash A \rangle \text{ type} \]

A term \( f : \langle \Phi \vdash A \rangle \) defines
Extension types

Formation rule for extension types

\[
\Phi \subset \Psi \quad \text{shape} \quad A \quad \text{type} \quad a : \Phi \to A
\]

\[
\langle \Phi \xrightarrow{a} \Psi \xrightarrow{} A \rangle \quad \text{type}
\]

A term \( f : \langle \Phi \xrightarrow{a} \Psi \xrightarrow{} A \rangle \) defines \( f : \Psi \to A \) so that \( f(t) \equiv a(t) \) for \( t : \Phi \).
Extension types

Formation rule for extension types

\[
\Phi \subset \Psi \quad \text{shape} \quad A \quad \text{type} \quad a : \Phi \to A
\]

\[\langle \Phi \ni a \to A \rangle \quad \text{type}\]

A term \(f : \langle \Phi \ni a \to A \rangle\) defines

\[f : \Psi \to A\] so that \(f(t) \equiv a(t)\) for \(t : \Phi\).

The simplicial type theory allows us to prove equivalences between extension types along composites or products of shape inclusions.
The synthetic theory of $\infty$-categories
The **hom type** for $A$ depends on two terms in $A$:

$$x, y : A \vdash \text{hom}_A(x, y)$$
The **hom type** for $A$ depends on two terms in $A$:

$$x, y : A \vdash \text{hom}_A(x, y)$$
Hom types

The **hom type** for \( A \) depends on two terms in \( A \):

\[
x, y : A \vdash \text{hom}_A(x, y)
\]

\[
\partial \Delta^1 \subset \Delta^1 \quad \text{shape} \quad A \text{ type} \quad [x, y] : \partial \Delta^1 \to A
\]

\[
\text{hom}_A(x, y) := \left< \begin{array}{c}
\partial \Delta^1 \\
\downarrow \\
\Delta^1 \\
\end{array} \xymatrix{\text{[x,y]} \ar[r] & A} \right> \text{ type}
\]

A term \( f : \text{hom}_A(x, y) \) defines an **arrow** in \( A \) from \( x \) to \( y \).
Segal types $\equiv$ types with binary composition

A type $A$ is **Segal** iff every composable pair of arrows has a unique composite.
A type $A$ is **Segal** iff every composable pair of arrows has a unique composite, i.e., for every $f : \text{Hom}_A(x, y)$ and $g : \text{Hom}_A(y, z)$ the type

$$\langle \Lambda^2_1 \xrightarrow{[f,g]} A \rangle$$

is contractible.
Segal types $\equiv$ types with binary composition

A type $A$ is **Segal** iff every composable pair of arrows has a unique composite, i.e., for every $f : \text{Hom}_A(x, y)$ and $g : \text{Hom}_A(y, z)$ the type $\langle \Lambda^2_1 \leftarrow \Delta^2, [f, g] \rightarrow A \rangle$ is contractible.

By contractibility, $\langle \Lambda^2_1 \leftarrow \Delta^2, [f, g] \rightarrow A \rangle$ has a unique inhabitant.
Segal types $\equiv$ types with binary composition

A type $A$ is **Segal** iff every composable pair of arrows has a unique composite, i.e., for every $f : \text{Hom}_A(x, y)$ and $g : \text{Hom}_A(y, z)$ the type

$$\langle \Lambda^2_1 \xymatrix{ \Delta^2 \ar@{~>}[r]^{[f,g]} \ar@{_{(}->}[d] & A \ar@{_{(}->}[d] } \rangle$$

is contractible.

By contractibility, $\langle \Lambda^2_1 \xymatrix{ \Delta^2 \ar@{~>}[r]^{[f,g]} \ar@{_{(}->}[d] & A \ar@{_{(}->}[d] } \rangle$ has a unique inhabitant. Write $g \circ f : \text{Hom}_A(x, z)$ for its inner face, *the* composite of $f$ and $g$. 
Identity arrows

For any $x : A$, the constant function defines a term

$$id_x := \lambda t. x : \text{Hom}_A(x, x) := \langle \partial \Delta^1 \xrightarrow{[x,x]} A \rangle,$$

which we denote by $id_x$ and call the identity arrow.
Identity arrows

For any $x : A$, the constant function defines a term

$$id_x := \lambda t. x : \text{Hom}_A(x, x) := \left\langle \begin{array}{c} \partial \Delta^1 \\ \downarrow \\ \Delta^1 \end{array} \xrightarrow{[x,x]} A \right\rangle,$$

which we denote by $id_x$ and call the identity arrow.

For any $f : \text{Hom}_A(x, y)$ in a Segal type $A$, the term

$$\lambda (s, t). f(t) : \left\langle \begin{array}{c} \Lambda^2_1 \\ \downarrow \\ \Delta^2 \end{array} \xrightarrow{[id_x, f]} A \right\rangle$$

witnesses the unit axiom $f = f \circ id_x$. 
Associativity of composition

Let $A$ be a Segal type with arrows

\[ f : \text{Hom}_A(x, y), \quad g : \text{Hom}_A(y, z), \quad h : \text{Hom}_A(z, w). \]
Associativity of composition

Let $A$ be a Segal type with arrows

$$f : \text{Hom}_A(x, y), \quad g : \text{Hom}_A(y, z), \quad h : \text{Hom}_A(z, w).$$

Prop. $h \circ (g \circ f) = (h \circ g) \circ f.$
Associativity of composition

Let $A$ be a Segal type with arrows

\[ f : \text{Hom}_A(x, y), \quad g : \text{Hom}_A(y, z), \quad h : \text{Hom}_A(z, w). \]

Prop. \[ h \circ (g \circ f) = (h \circ g) \circ f. \]

Proof: Consider the composable arrows in the Segal type $\Delta^1 \rightarrow A$:
**Associativity of composition**

Let $A$ be a Segal type with arrows

\[ f : \text{Hom}_A(x, y), \quad g : \text{Hom}_A(y, z), \quad h : \text{Hom}_A(z, w). \]

Prop. \[ h \circ (g \circ f) = (h \circ g) \circ f. \]

Proof: Consider the composable arrows in the Segal type $\Delta^1 \to A$:

Composing defines a term in the type $\Delta^2 \to (\Delta^1 \to A)$
Associativity of composition

Let $A$ be a Segal type with arrows

\[ f : \text{Hom}_A(x, y), \quad g : \text{Hom}_A(y, z), \quad h : \text{Hom}_A(z, w). \]

Prop. \[ h \circ (g \circ f) = (h \circ g) \circ f. \]

Proof: Consider the composable arrows in the Segal type $\Delta^1 \to A$:

Composing defines a term in the type $\Delta^2 \to (\Delta^1 \to A)$ which yields a term $\ell : \text{Hom}_A(x, w)$ so that $\ell = h \circ (g \circ f)$ and $\ell = (h \circ g) \circ f$. 
Isomorphisms

An arrow \( f : \text{Hom}_A(x, y) \) in a Segal type is an isomorphism if it has a two-sided inverse \( g : \text{Hom}_A(y, x) \). However, the type

\[
\sum_{g : \text{Hom}_A(y, x)} (g \circ f = \text{id}_x) \times (f \circ g = \text{id}_y)
\]

has higher-dimensional structure and is not a proposition.
Isomorphisms

An arrow $f : \text{Hom}_A(x,y)$ in a Segal type is an isomorphism if it has a two-sided inverse $g : \text{Hom}_A(y,x)$. However, the type

$$\sum_{g : \text{Hom}_A(y,x)} (g \circ f = \text{id}_x) \times (f \circ g = \text{id}_y)$$

has higher-dimensional structure and is not a proposition. Instead define

$$\text{isiso}(f) := \left( \sum_{g : \text{Hom}_A(y,x)} g \circ f = \text{id}_x \right) \times \left( \sum_{h : \text{Hom}_A(y,x)} f \circ h = \text{id}_y \right).$$
Isomorphisms

An arrow $f : \text{Hom}_A(x, y)$ in a Segal type is an isomorphism if it has a two-sided inverse $g : \text{Hom}_A(y, x)$. However, the type

$$\sum_{g : \text{Hom}_A(y, x)} (g \circ f = \text{id}_x) \times (f \circ g = \text{id}_y)$$

has higher-dimensional structure and is not a proposition. Instead define

$$\text{isiso}(f) := \left( \sum_{g : \text{Hom}_A(y, x)} g \circ f = \text{id}_x \right) \times \left( \sum_{h : \text{Hom}_A(y, x)} f \circ h = \text{id}_y \right).$$

For $x, y : A$, the type of isomorphisms from $x$ to $y$ is:

$$x \cong_A y := \sum_{f : \text{Hom}_A(x, y)} \text{isiso}(f).$$
Rezk types $\equiv \infty$-categories

By path induction, to define a map

$$\text{path-to-iso} : (x =_A y) \to (x \simeq_A y)$$

for all $x, y : A$ it suffices to define

$$\text{path-to-iso}(\text{refl}_x) := \text{id}_x.$$
Rezk types ≡ ∞-categories

By path induction, to define a map

\[ \text{path-to-iso} : (x =_A y) \rightarrow (x \simeq_A y) \]

for all \( x, y : A \) it suffices to define

\[ \text{path-to-iso}(\text{refl}_x) := \text{id}_x. \]

A Segal type \( A \) is Rezk iff every isomorphism is an identity
Rezk types $\equiv \infty$-categories

By path induction, to define a map

$$\text{path-to-iso} : (x =_A y) \to (x \cong_A y)$$

for all $x, y : A$ it suffices to define

$$\text{path-to-iso}(\text{refl}_x) := \text{id}_x.$$

A Segal type $A$ is Rezk iff every isomorphism is an identity, i.e., iff the map

$$\text{path-to-iso} : \prod_{x, y : A} (x =_A y) \to (x \cong_A y)$$

is an equivalence.
Discrete types $\equiv$ $\infty$-groupoids

Similarly by path induction define

$$\text{path-to-arr}: (x =_A y) \to \text{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{path-to-arr}(\text{refl}_x) := \text{id}_x$. 
Discrete types $\equiv \infty$-groupoids

Similarly by path induction define

$$\text{path-to-arr}: (x \equiv_A y) \to \text{Hom}_A(x, y)$$

for all $x, y : A$ by $\text{path-to-arr}(\text{refl}_x) := \text{id}_x$.

A type $A$ is discrete iff every arrow is an identity, i.e., iff $\text{path-to-arr}$ is an equivalence.
Discrete types ≡ ∞-groupoids

Similarly by path induction define

\[ \text{path-to-arr}: (x =_A y) \to \text{Hom}_A(x, y) \]

for all \( x, y : A \) by \( \text{path-to-arr}(\text{refl}_x) := \text{id}_x \).

A type \( A \) is discrete iff every arrow is an identity, i.e., iff \( \text{path-to-arr} \) is an equivalence.

Prop. A type is discrete if and only if it is Rezk and all of its arrows are isomorphisms.

Proof:

\[
\begin{array}{ccc}
  x & \xrightarrow{\text{path-to-arr}} & \text{Hom}_A(x, y) \\
  \downarrow^{\text{path-to-iso}} & & \downarrow \\
  x \cong_A y
\end{array}
\]
\(\infty\)-categories for undergraduates

defn. An \(\infty\)-groupoid is a type in which arrows are equivalent to identities:

\[
\text{path-to-arr}: (x =_A y) \rightarrow \text{Hom}_A(x, y) \text{ is an equivalence.}
\]
defn. An $\infty$-groupoid is a type in which arrows are equivalent to identities:

$$\text{path-to-arr}: (x =_A y) \rightarrow \text{Hom}_A(x, y)$$ is an equivalence.

defn. An $\infty$-category is a type

- which has unique binary composites of arrows:

$$\langle \Lambda^2_1 \xrightarrow{[f,g]} A \rangle$$ is contractible
∞-categories for undergraduates

defn. An ∞-groupoid is a type in which arrows are equivalent to identities:

\[ \text{path-to-arr}: \ (x \cong_A y) \to \text{Hom}_A(x, y) \text{ is an equivalence.} \]

defn. An ∞-category is a type

- which has unique binary composites of arrows:

\[
\begin{array}{c}
\Lambda^2 \xrightarrow{[f, g]} A \\
\Delta^2 \xleftarrow{} \\
\end{array}
\]

is contractible

- and in which isomorphisms are equivalent to identities:

\[ \text{path-to-iso}: \ (x \cong_A y) \to (x \cong_A y) \text{ is an equivalence.} \]
References

For considerably more, see:

Emily Riehl and Michael Shulman

To explore homotopy type theory:


Michael Shulman, Homotopy type theory: the logic of space, arXiv:1703.03007

Thank you!