Quillen Model Categories
Model
Martin-Löf Type Theory with Identity Types

Samuel Mimram

CEA

Séminaire MeASI au CIRM
Ideas are not from me (Awodey & Warren, Voevodsky, ...), errors are mine.
\textbf{\(\lambda\)-calculus}

- **Introduction rule:**

\[
\frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x.f : A \to B}
\]
• Introduction rule:

\[ \Gamma, x: A \vdash f : B \]

\[ \Gamma \vdash \lambda x. f : A \rightarrow B \]

• Elimination rule:

\[ \Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash g : A \]

\[ \Gamma \vdash fg : B \]
• Introduction rule:

\[ \Gamma, x : A \vdash f : B \]
\[ \Gamma \vdash \lambda x.f : A \rightarrow B \]

• Elimination rule:

\[ \Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash g : A \]
\[ \Gamma \vdash fg : B \]

• Conversion rule:

\[ \Gamma, x : A \vdash f : B \quad \Gamma \vdash g : A \]
\[ \Gamma \vdash (\lambda x.f)g = f[g/x] : B \]
Now with dependent types.
Dependent types

Array.make : int -> array
Dependent types

Array.make : \( n : \text{int} \rightarrow n \text{ array} \)
Dependent types

Array.make : $\Pi_{n:\text{int}} \text{array}(n)$
Dependent types

Array.make : \( \Pi_{n: \text{int}} \text{array}(n) \)

- Type formation rule:

\[
\frac{\vdash n : \text{int}}{\vdash \text{array}(n) : \text{type}}
\]
Dependent types

Array.make : $\Pi_{n : \text{int}} \text{array}(n)$

- Type formation rule:

$$
\vdash n : \text{int} \\
\vdash \text{array}(n) : \text{type}
$$

- Introduction rules:

$$
\Gamma \vdash [] : \text{array}(0) \\
\Gamma, n : \text{int} \vdash k : \text{int} \\
\Gamma, n : \text{int} \vdash a : \text{array}(n) \\
\Gamma, n : \text{int} \vdash (k :: a) : \text{array}(n + 1)
$$
Products (and sums)

- Type formation rule:

\[ x : A \vdash B(x) : \text{type} \]
\[ \vdash \Pi_{x : A} B(x) : \text{type} \]
Products (and sums)

- **Type formation rule:**

\[
\begin{align*}
\Gamma : A & \vdash B(x) : \text{type} \\
\Gamma & \vdash \prod_{x : A} B(x) : \text{type}
\end{align*}
\]

- **Introduction rule:**

\[
\begin{align*}
\Gamma : A & \vdash f(x) : B(x) \\
\Gamma & \vdash \lambda_{x : A} f(x) : \prod_{x : A} B(x)
\end{align*}
\]
Products (and sums)

- Type formation rule:
  \[ x : A \vdash B(x) : \text{type} \]
  \[ \vdash \Pi_{x : A} B(x) : \text{type} \]

- Introduction rule:
  \[ x : A \vdash f(x) : B(x) \]
  \[ \vdash \lambda_{x : A} f(x) : \Pi_{x : A} B(x) \]

- Elimination rule:
  \[ \vdash g : \Pi_{x : A} B(x) \quad \vdash x : A \]
  \[ \vdash ga : B(a) \]
Products (and sums)

- **Type formation rule:**

  \[ x : A \vdash B(x) : \text{type} \]
  \[ \vdash \Pi_{x:A} B(x) : \text{type} \]

- **Introduction rule:**

  \[ x : A \vdash f(x) : B(x) \]
  \[ \vdash \lambda x:A. f(x) : \Pi_{x:A} B(x) \]

- **Elimination rule:**

  \[ \vdash g : \Pi_{x:A} B(x) \]
  \[ \vdash x : A \]
  \[ \vdash g a : B(a) \]

- **Conversion rule:**

  \[ x : A \vdash f(x) : B(x) \]
  \[ \vdash a : A \]
  \[ \vdash (\lambda x:A.f(x)) a = f(a) : B(a) \]
Remark

The usual arrow type $A \to B$ is recovered as

$$\Pi_{x:A} B$$

where $x$ does not occur in $B$. 
Identity types

- Type formation rule:

\[ \vdash a : A \quad \vdash b : A \]

\[ \vdash \text{Id}_A(a, b) : \text{type} \]
Identity types

• Type formation rule:

\[ \vdash a : A \quad \vdash b : A \]
\[ \vdash \text{Id}_A(a, b) : \text{type} \]

• Introduction rule:

\[ \vdash a : A \]
\[ \vdash \text{r}_A(a) : \text{Id}_A(a, a) \]
Identity types

• Type formation rule:

\[ \vdash a : A \quad \vdash b : A \]
\[ \vdash \text{Id}_A(a, b) : \text{type} \]

• Introduction rule:

\[ \vdash a : A \]
\[ \vdash \text{r}_A(a) : \text{Id}_A(a, a) \]

• Elimination rule:

\[ x : A, y : A, z : \text{Id}_A(x, y) \vdash D(x, y, z) : \text{type} \]
\[ \vdash p : \text{Id}_A(a, b) \quad x : A \vdash d(x) : D(x, x, \text{r}_A(x)) \]
\[ \vdash J_{A,D}(d, a, b, p) : D(a, b, p) \]
Identity types

- Type formation rule:
  \[ \vdash a : A \quad \vdash b : A \]
  \[ \vdash \text{Id}_A(a, b) : \text{type} \]

- Introduction rule:
  \[ \vdash a : A \]
  \[ \vdash r_A(a) : \text{Id}_A(a, a) \]

- Elimination rule:
  \[ x : A, y : A, z : \text{Id}_A(x, y) \vdash D(x, y, z) : \text{type} \]
  \[ \vdash p : \text{Id}_A(a, b) \quad x : A \vdash d(x) : D(x, x, r_A(x)) \]
  \[ \vdash J_{A,D}(d, a, b, p) : D(a, b, p) \]

- Conversion rule:
  \[ x : A, y : A, z : \text{Id}_A(x, y) \vdash D(x, y, z) : \text{type} \]
  \[ \vdash a : A \quad x : A \vdash d(x) : D(x, x, r_A(x)) \]
  \[ \vdash J_{A,D}(d, a, a, r_A(a)) = d(a) : D(a, a, r_A(a)) \]
A category $\mathcal{C}$ consists of

- objects: $\text{Ob}(\mathcal{C})$
- morphisms: $\forall A, B \in \text{Ob}(\mathcal{C}), \quad \text{Hom}(A, B)$
- compositions:

  \[
  \begin{array}{ccc}
  f : A \to B & g : B \to C & g \circ f : A \to C \\
  \end{array}
  \]

- identities:

  $\forall A \in \text{Ob}(\mathcal{C}), \quad \text{id}_A : A \to A$

such that

- composition is associative:

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

- admits identities as neutral elements

  \[
  \text{id} \circ f = f = f \circ \text{id}
  \]
The category \textbf{Set} has

- objects: sets
- morphisms: functions $f : A \rightarrow B$
- with usual composition and identities
Modeling programming languages

From a programming language, we can build a category $\Pi$ whose

- objects: types
- morphisms: programs $\pi : A \to B$ modulo cut-elimination
- composition: usual composition of programs
Modeling programming languages

From a programming language, we can build a category $\Pi$ whose

- objects: types
- morphisms: programs $\pi : A \to B$ modulo cut-elimination
- composition: usual composition of programs

Definition

A **model** of the programming language is a functor

$$F : \Pi \to C$$
Models of simply typed $\lambda$-calculus

Take the category with

- objects: types

\[
A ::= X \mid A \Rightarrow B \mid A \times B
\]

- morphisms $A \to B$: $\lambda$-terms $f : A \Rightarrow B$
Models of simply typed $\lambda$-calculus

Take the category with
- objects: types

\[ A ::= X \mid A \Rightarrow B \mid A \times B \]

- morphisms $A \to B$: $\lambda$-terms $f : A \Rightarrow B$

Example:

\[ \lambda x.\lambda y. x : A \to (B \Rightarrow A) \]
Models of simply typed $\lambda$-calculus

Take the category with

- objects: types
  
  $A ::= X \mid A \Rightarrow B \mid A \times B$

- morphisms $A \to B$: $\lambda$-terms $f : A \Rightarrow B$

Exercise: give a model of this language into $\text{Set}$. 
Models of simply typed $\lambda$-calculus

Take the category with

- objects: types

\[
A ::= X \mid A \Rightarrow B \mid A \times B
\]

- morphisms $A \rightarrow B$: $\lambda$-terms $f : A \Rightarrow B$

More generally, it can be modeled in any cartesian closed category.
Definition

A cartesian closed category is a category which has

- products:

\[ \forall f : A \to B, g : A \to C, \]

\[ \begin{array}{ccc}
A & \xleftarrow{f} & B \times C \\
& \searrow & \\
& f & \searrow \\
& & B \\
& \downarrow & \\
& \pi_B & \\
& \downarrow & \\
& & C \\
& \downarrow & \\
& \pi_C & \\
& \downarrow & \\
& & C
\end{array} \]
Cartesian closed categories

Definition

A **cartesian closed category** is a category which has

- **products:**

\[ \forall f : A \to B, g : A \to C, \]

- **a terminal object** 1:

\[ \forall A, A \to 1 \]
Cartesian closed categories

Definition

A cartesian closed category is a category which has

- products:

\[ \forall f : A \to B, g : A \to C, \]

- a terminal object 1:

\[ \forall A, \quad A \to 1 \]

- which is closed:

\[
\begin{align*}
A \times B & \to C \\
\frac{A \to (B \Rightarrow C)}{}
\end{align*}
\]
A model of Martin-Löf type theory

The traditional models of Martin-Löf type theory are given by

**Definition**

A *locally cartesian closed category* is a category $\mathcal{C}$ in which for every object $A$ the slice category $\mathcal{C}/A$ is cartesian closed.
A model of Martin-Löf type theory

The traditional models of Martin-Löf type theory are given by

**Definition**

A locally cartesian closed category is a category $C$ in which for every object $A$ the slice category $C/A$ is cartesian closed.

**Theorem**

An LCCC is a category with pullbacks in which for every $f : A \to B$, the base change functor $f^* : C/B \to C/A$ has a right adjoint $\Pi_f : C/A \to C/B$. 
A model of Martin-Löf type theory

The traditional models of Martin-Löf type theory are given by

Definition

A **locally cartesian closed category** is a category $\mathcal{C}$ in which for every object $A$ the slice category $\mathcal{C}/A$ is cartesian closed.

Theorem

An LCCC is a category with pullbacks in which for every $f : A \to B$, the base change functor $f^* : \mathcal{C}/B \to \mathcal{C}/A$ has a right adjoint $\Pi_f : \mathcal{C}/A \to \mathcal{C}/B$.

Example

\[
\Gamma, x : A \vdash B(x) : \text{type} \\
\Gamma \vdash \Pi_{x:A}.B(x) : \text{type}
\]
Every LCCC is also a model of MLTT with the rule of extensionality:

\[ \vdash p : \text{Id}_A(a, b) \]
\[ \vdash a = b : A \]

...and type checking is indecidable in extensional MLTT!
We explain here that Quillen model categories model identity types in Martin-Löf type theory:

\[ F : \mathcal{M} \to \mathcal{Q} \]

Which provides non-extensional models.
We explain here that Quillen model categories model identity types in Martin-Löf type theory:

\[ F : \mathcal{M} \to \mathcal{Q} \]

Which provides non-extensional models.

The idea here is that identity types behave like homotopies between topological spaces.
Homotopy

A homotopy between two continuous functions $f, g : A \to B$ between topological spaces $A$ and $B$ is a continuous function

$$h : I \times A \to B$$

where $I = [0, 1]$ such that $h(0, x) = f(x)$ and $h(1, x) = g(x)$. 

Two spaces $A$ and $B$ are homotopy equivalent when there exist maps $f : A \to B$ and $g : B \to A$ such that $g \circ f \sim \text{id}_A$ and $f \circ g \sim \text{id}_B$. 

Ex: square $\approx$ circle, coffee mug $\approx$ donut, etc.
A homotopy between two continuous functions $f, g : A \to B$ between topological spaces $A$ and $B$ is a continuous function

$$h : I \times A \to B$$

where $I = [0, 1]$ such that $h(0, x) = f(x)$ and $h(1, x) = g(x)$.

Two spaces $A$ and $B$ are **homotopy equivalent** when there exists maps $f : A \to B$ and $g : B \to A$ such that

$$g \circ f \sim \text{id}_A \quad f \circ g \sim \text{id}_B$$

Ex: square $\approx$ circle, coffee mug $\approx$ donut, etc.
Homotopies

Suppose given a topological space $T$.

- A **path** in $T$ is a continuous function $\pi : I \rightarrow T$, where $I = [0, 1]$. 
Homotopies

Suppose given a topological space $T$.

- A **path** in $T$ is a continuous function $\pi : I \to T$, where $I = [0, 1]$.

- An **homotopy** between two paths $\pi$ and $\rho$ is a continuous function

$$h : I \to (I \Rightarrow T)$$

such that $h(0) = \pi$ and $h(1) = \rho$.
Homotopies

Suppose given a topological space $T$.

- A **path** in $T$ is a continuous function $\pi : I \to T$, where $I = [0, 1]$.

- An **homotopy** between two paths $\pi$ and $\rho$ is a continuous function

$$h : I \to (I \Rightarrow T) \quad \text{such that} \quad h(0) = \pi \quad \text{and} \quad h(1) = \rho$$

- An **homotopy between homotopies** $h$ and $k$ is a continuous function

$$h : I \to (I \Rightarrow (I \Rightarrow T)) \quad \text{such that} \quad h(0) = h \quad \text{and} \quad h(1) = k$$
Homotopies

Suppose given a topological space $T$.

- A **path** in $T$ is a continuous function $\pi : I \to T$, where $I = [0, 1]$.
- An **homotopy** between two paths $\pi$ and $\rho$ is a continuous function
  \[ h : I \to (I \Rightarrow T) \] such that $h(0) = \pi$ and $h(1) = \rho$
- An **homotopy between homotopies** $h$ and $k$ is a continuous function
  \[ h : I \to (I \Rightarrow (I \Rightarrow T)) \] such that $h(0) = h$ and $h(1) = k$
- etc.
We interpret

- a type $\vdash A : \text{type}$ as a topological space
- a term $\vdash x : A$ as a point in $A$
- a term $p : \text{Id}_A(a,b)$ as a path $a \to b$
- a term $s : \text{Id}_{\text{Id}(a,b)}(p,q)$ as an homotopy $a \xrightarrow{p} b$ \xrightarrow{s} q \xrightarrow{t} a$
- etc.
Dependent types

As in the case of LCCC we interpret a dependent type

\[ x : A \vdash B(x) : \text{type} \]

as a continuous map

\[
\begin{array}{c}
\text{B} \\
\downarrow \\
\text{A}
\end{array}
\]
Dependent types

As in the case of LCCC we interpret a dependent type

\[ x : A \vdash B(x) : \text{type} \]

as a continuous map

\[ \begin{array}{c}
B \downarrow \\
\downarrow \\
A
\end{array} \]

and a term \( x : A \vdash f : B(x) \) as a section of this map.
Dependent types and equality

The maps interpreting types should have the \textit{homotopy lifting} property:

\[
\begin{array}{ccc}
\{0\} & \xrightarrow{\beta} & B \\
\downarrow & & \downarrow \\
[0,1] & \xrightarrow{p^*} & A
\end{array}
\]

Maps like this are often called \textit{fibrations}.

Ex: the interpretation of \(x, y: A \vdash \text{Id}_A(x, y)\) is a map

\[
\begin{array}{ccc}
A & \downarrow & A \\
\downarrow & & \downarrow \\
A & \xrightarrow{p} & A
\end{array}
\]
Dependent types and equality

The maps interpreting types should have the homotopy lifting property:

\[
\begin{array}{ccc}
X \times \{0\} & \xrightarrow{\beta} & B \\
\downarrow & & \downarrow \\
X \times [0, 1] & \xrightarrow{p} & A
\end{array}
\]

Maps like this are often called fibrations.
Dependent types and equality

The maps interpreting types should have the homotopy lifting property:

\[
\begin{array}{c}
X \times \{0\} \xrightarrow{\beta} B \\
\downarrow \quad \downarrow \\
X \times [0, 1] \xrightarrow{p^*} A
\end{array}
\]

Maps like this are often called fibrations.

Ex: the interpretation of \( x, y : A \vdash \text{Id}_A(x, y) \) is a map

\[
\begin{array}{c}
A^I \\
\downarrow \\
A \times A
\end{array}
\]
Lifting properties

Homotopy is more generally carried on in Quillen model categories.
Lifting properties

Homotopy is more generally carried on in Quillen model categories.

Definition

Given maps \( f : A \to B \) and \( g : C \to D \), \( f \) has the left lifting property \( \text{wrt } g \) when every commutative square

\[
\begin{array}{c}
A \\
\downarrow f \\
B \\
\hline
\end{array}
\begin{array}{c}
/ C \\
\downarrow g \\
/ D \\
\hline
\end{array}
\begin{array}{c}
A \\
\downarrow h \\
C \\
\hline
B \\
\end{array}
\]

admits a lifting.

Given a class \( L \) of maps, we write \( \bot L \) for the class of maps which have LLP \( \text{wrt every map in } L \) (and similarly \( L \bot \) for RLP).
Lifting properties

Homotopy is more generally carried on in Quillen model categories.

Definition
Given maps $f : A \rightarrow B$ and $g : C \rightarrow D$, $f$ has the left lifting property wrt $g$ when every commutative square admits a lifting.

\[
\begin{array}{ccc}
A & \xrightarrow{h} & C \\
\downarrow{f} & & \downarrow{g} \\
B & \xleftarrow{k} & D
\end{array}
\]

Given a class $L$ of maps, we write $\perp L$ for the class of maps which have LLP wrt every map in $L$ (and similarly $L \perp$ for RLP).
Lifting properties

Homotopy is more generally carried on in Quillen model categories.

Definition

Given maps \( f : A \to B \) and \( g : C \to D \), \( f \) has the left lifting property wrt \( g \) when every commutative square

\[
\begin{array}{ccc}
A & \xrightarrow{h} & C \\
\downarrow{f} & & \downarrow{g} \\
B & \xleftarrow{l} & D
\end{array}
\]

admits a lifting.

Given a class \( \mathcal{L} \) of maps, we write \( \perp \mathcal{L} \) for the class of maps which have LLP wrt every map in \( \mathcal{L} \) (and similarly \( \mathcal{L} \perp \) for RLP).
Definition

A weak factorization system \((\mathcal{L}, \mathcal{R})\) consists of two classes of maps such that

1. every map \(f : A \to B\) factors as

\[
\begin{array}{c}
A \\
\downarrow f \\
\downarrow i \\
\downarrow p \\
B
\end{array}
\]

with \(i \in \mathcal{L}\) and \(p \in \mathcal{R}\)

2. \(\mathcal{L}^\perp = \mathcal{R}\) and \(\mathcal{L} = \perp \mathcal{R}\)
Model categories

Definition
A model category consists of $C$ together with subcategories
- $\mathcal{F}$: fibrations
- $\mathcal{E}$: cofibrations
- $\mathcal{W}$: weak equivalences
such that
1. three for two
2. both $(\mathcal{E}, \mathcal{W} \cap \mathcal{F})$ and $(\mathcal{C} \cap \mathcal{W}, \mathcal{F})$ are weak factorization systems.

Example
On Top:
- generating cofibrations are inclusions $i: \Delta^n \to \Delta^n \times I$,
- fibrations are RLP of generating cofibrations (Serre fibrations),
- weak equivalences are weak homotopy equivalences.
Model categories

Definition

A model category consists of $C$ together with subcategories

- $\mathcal{F}$: fibrations
- $\mathcal{C}$: cofibrations
- $\mathcal{W}$: weak equivalences

such that

1. three for two
2. both $(\mathcal{C}, \mathcal{W} \cap \mathcal{F})$ and $(\mathcal{C} \cap \mathcal{W}, \mathcal{F})$ are weak factorization systems.

Example

On Top:

- generating cofibrations are inclusions $i : \Delta^n \to \Delta^n \times I$,
- fibrations are RLP of generating cofibrations (Serre fibrations),
- weak equivalences are weak homotopy equivalences.
Definition

A (very good) path object $A^I$ for an object $A$ consists of a factorization

\[
\begin{array}{ccc}
A & \xrightarrow{r} & A^I \\
\downarrow{\Delta} & & \downarrow{\Delta} \\
A \times A & \xleftarrow{p} & A^I
\end{array}
\]

with $r$ acyclic cofibration and $p$ fibration.
Interpretation of MLTT

\[
\begin{align*}
A & \vdash \sqrt{A'} \\
A \times A & \vdash \Delta \\
A & \vdash r
\end{align*}
\]
Interpretation of MLTT

A → → ∆↘↘A

A × A

• Type formation rule:

\[ \vdash a : A \quad \vdash b : A \]

\[ \vdash \text{Id}_A(a, b) : \text{type} \]

\(\text{Id}_A\) is interpreted as \(\_\)
Interpretation of MLTT

- Type formation rule:
- Introduction rule:

\[ \vdash a : A \]

\[ \vdash r_A(a) : \text{Id}_A(a, a) \]

\(r_A\) is interpreted as \(r\)
Interpretation of MLTT

- **Type formation rule:**
- **Introduction rule:**
- **Elimination rule:**

\[
\begin{align*}
    x : A, y : A, z : \text{Id}_A(x, y) & \vdash D(x, y, z) : \text{type} \\
    x : A & \vdash d(x) : D(x, x, r_A(x)) \\
    x : A, y : A, z : \text{Id}_A(x, y) & \vdash J_{A, D}(d, x, y, z) : D(x, y, z)
\end{align*}
\]
Interpretation of MLTT

- Type formation rule:
- Introduction rule:
- Elimination rule:

\[
\begin{align*}
& \text{x : A, y : A, z : Id_A(x, y) \vdash D(x, y, z) : type} \\
& \quad x : A \vdash d(x) : D(x, x, r_A(x)) \\
& \quad x : A, y : A, z : Id_A(x, y) \vdash J_{A,D}(d, x, y, z) : D(x, y, z)
\end{align*}
\]

- Conversion rule:

\[
\begin{align*}
& \text{x : A, y : A, z : Id_A(x, y) \vdash D(x, y, z) : type} \\
& \quad x : A \vdash d(x) : D(x, x, r_A(x)) \\
& \quad x : A \vdash J_{A,D}(d, x, x, r_A(x)) = d(x) : D(x, x, r_A(x))
\end{align*}
\]
The current state of things

Theorem (Awodey & Warren)
MLTT can be interpreted in any model category.

Theorem (Gambino & Garner)
The interpretation is complete.
The Homotopy Hypothesis

Homotopy Types $\xrightarrow{\text{??}}$ Weak $\omega$-groupoids

MLTT

$\text{??}$
Towards directed algebraic topology?

We could think of a directed variant:

- replace equality by a reduction relation:

\[ f \rightsquigarrow g \Rightarrow \text{there is a directed path from } f \text{ to } g \]
Towards directed algebraic topology?

We could think of a directed variant:

- replace equality by a reduction relation:
  \[ f \leadsto g \implies \text{there is a directed path from } f \text{ to } g \]

- the reduction should be compatible with identity:
  \[ r : \text{Id}(f,f') \implies \exists g', \exists s : \text{Id}(g,g') \text{ and } g \leadsto g' \]

We can “translate continuously” the directed path \( f \leadsto g \) into the directed path \( f' \leadsto g \).