Martin-Löf Type theory

Anja Petković Komel

OPLSS 2025

- In type theory every element comes equipped with a type (a : A)

- Type theory = deductive system
  Inference rules make derivations

- We derive **judgements**.

$$\frac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \ldots \quad \mathcal{H}_n}{\mathcal{C}.}$$

Inference rules

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash f : A \to B}{\Gamma \vdash f(a) : B.}$$

Inference rule for function types

There are 4 kinds of **judgements**:

context judgement thesis

$\Gamma \vdash A \text{ type}.$

well-formed type

$\Gamma \vdash A \doteq B \text{ type}.$

judgementally equal types

$\Gamma \vdash a : A.$

well-formed term

$\Gamma \vdash a \doteq b : A.$

judgementally equal terms

A context $x_1 : A_1,\ x_2 : A_2(x_1),\ \ldots,\ x_n : A_n(x_1, \ldots, x_{n-1})$ such that

$$x_1 : A_1,\ \ldots,\ x_{k-1} : A_{k-1}(x_1, \ldots, x_{k-2}) \vdash A_k(x_1, \ldots, x_{k-1})\ \text{type},$$

A context of length 0: **empty context**

Dependent types and terms

$\Gamma,\ x : A \vdash B(x)\ \text{type},$ type family $\quad n : \mathbb{N} \vdash \text{Vec}\ \mathbb{N}\ n\ \ \text{type}$

$\Gamma,\ x : A \vdash b(x) : B(x)$ section $\quad n : \mathbb{N} \vdash (0,0,\ldots,0) : \text{Vec}\ \mathbb{N}\ n$

n-times

Inference rules

Structural rules (in every type theory)

Specific rules (for MLTT)

Structural rules

- Judgemental equality is an equivalence relation
- Variable conversion
- Substitution
- Weakening
- Generic element / variable rule

Judgemental equality is an equivalence relation

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \doteq A \text{ type}} \qquad \frac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash B \doteq A \text{ type}} \qquad \frac{\Gamma \vdash A \doteq B \text{ type} \qquad \Gamma \vdash B \doteq C \text{ type}}{\Gamma \vdash A \doteq C \text{ type}}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \doteq a : A} \qquad \frac{\Gamma \vdash a \doteq b : A}{\Gamma \vdash b \doteq a : A} \qquad \frac{\Gamma \vdash a \doteq b : A \qquad \Gamma \vdash b \doteq c : A}{\Gamma \vdash a \doteq c : A.}$$

Variable conversion

$$\frac{\Gamma \vdash A \doteq A' \text{ type} \qquad \Gamma,\ x : A,\ \Delta \vdash B(x) \text{ type}}{\Gamma,\ x : A',\ \Delta \vdash B(x) \text{ type}.}$$

element conversion is derivable

$$\frac{\Gamma \vdash A \doteq A' \text{ type} \qquad \Gamma \vdash a : A}{\Gamma \vdash a : A'.}$$

Substitution

$$\frac{\Gamma \vdash a : A \qquad \Gamma,\ x : A,\ \Delta \vdash \mathcal{J}}{\Gamma,\ \Delta[a/x] \vdash \mathcal{J}[a/x]}\ S.$$

$$\frac{\vdash 0 : \mathbb{N} \qquad n : \mathbb{N} \vdash \mathrm{succ}(n) : \mathbb{N}}{\vdash \mathrm{succ}(0) : \mathbb{N}}$$

$$\frac{\Gamma \vdash a \doteq a' : A \qquad \Gamma,\ x : A,\ \Delta \vdash B\ \mathrm{type}}{\Gamma,\ \Delta[a/x] \vdash B[a/x] \doteq B[a'/x]\ \mathrm{type}}$$

$$\frac{\Gamma \vdash a \doteq a' : A \qquad \Gamma,\ x : A,\ \Delta \vdash b : B}{\Gamma,\ \Delta[a/x] \vdash b[a/x] \doteq b[a'/x] : B[a/x].}$$

substituting judgementally equal stuff results in judgementally equal types (resp. terms)

Weakening

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma, \Delta \vdash \mathcal{J}}{\Gamma, x : A, \Delta \vdash \mathcal{J}}\, W.$$

why is this wrong?

$$\frac{\vdash \mathbb{N} \text{ type} \qquad x : \mathbb{N} \vdash \text{succ}(0) : \mathbb{N}}{x : \mathbb{N}\,,\, y : \mathbb{N} \vdash \text{succ}(0) : \mathbb{N}}$$

Generic element / variable rule

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}\, \delta.$$

$$\frac{\vdash \mathbb{N} \text{ type}}{x : \mathbb{N} \vdash x : \mathbb{N}}$$

Structural rules

- Judgemental equality is an equivalence relation
- Variable conversion
- Substitution
- Weakening
- Generic element / variable rule

Derivations

A derivation is a finite tree:
- nodes are inferences
- root is conclusion
- leaves are hypotheses

$$\frac{\Gamma \vdash A \doteq A' \text{ type} \qquad \Gamma \vdash a : A}{\Gamma \vdash a : A'.}$$

$$\dfrac{\dfrac{\Gamma \vdash a : A \qquad \dfrac{\dfrac{\Gamma \vdash A \doteq A' \text{ type}}{\Gamma \vdash A' \doteq A \text{ type}} \qquad \dfrac{\Gamma \vdash A' \text{ type}}{\Gamma, x : A' \vdash x : A'}}{\Gamma, x : A \vdash x : A'}}{\Gamma \vdash a : A'}$$

Specific rules

Dependent functions

Functions where type of the output may depend on the input.

Dependent function type:

$$\prod_{(x:A)} B(x)$$

Rules for Π types:

- formation rule — form types
- introduction rule — introduce terms
- elimination rule — use terms
- computation rules — interaction
- congruence rules — respect judgemental equality

Dependent functions

Formation rule

$$\frac{\Gamma, x : A \vdash B(x)\ \text{type}}{\Gamma \vdash \prod_{(x:A)} B(x)\ \text{type}}\ \Pi.$$

Introduction rule (lambda)

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x.\, b(x) : \prod_{(x:A)} B(x)}\ \lambda.$$

binding

$$\frac{\Gamma \vdash A \doteq A'\ \text{type} \qquad \Gamma, x : A \vdash B(x) \doteq B'(x)\ \text{type}}{\Gamma \vdash \prod_{(x:A)} B(x) \doteq \prod_{(x:A')} B'(x)\ \text{type}}\ \Pi\text{-eq.}$$

$$\frac{\Gamma, x : A \vdash b(x) \doteq b'(x) : B(x)}{\Gamma \vdash \lambda x.\, b(x) \doteq \lambda x.\, b'(x) : \prod_{(x:A)} B(x)}\ \lambda\text{-eq.}$$

$$\frac{x : \mathbb{N} \vdash \text{Vec } \mathbb{N}\ x \quad \text{type}}{\vdash \Pi\ (x : \mathbb{N})\ \text{Vec } \mathbb{N}\ x \quad \text{type}}$$

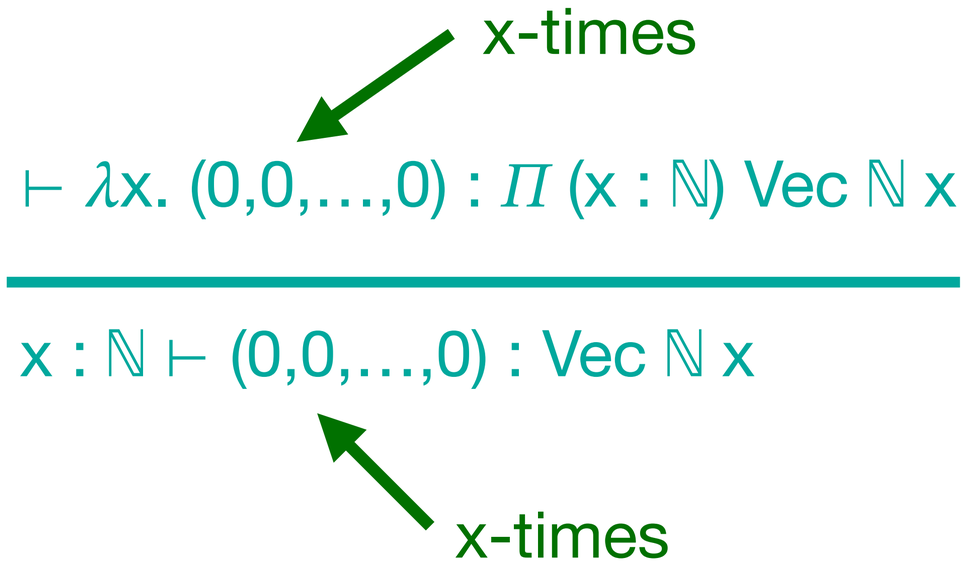
$$\frac{x : \mathbb{N} \vdash (0,0,\ldots,0) : \text{Vec } \mathbb{N}\ x}{\vdash \lambda x.\ (0,0,\ldots,0) : \Pi\ (x : \mathbb{N})\ \text{Vec } \mathbb{N}\ x}$$

x-times

Dependent functions

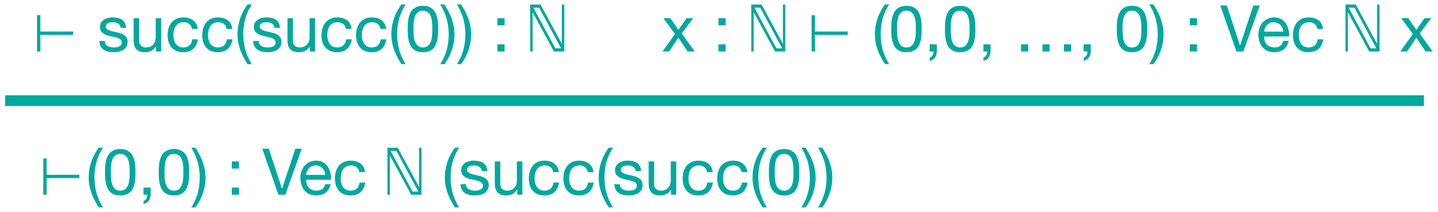
Elimination (evaluation) rule

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma, x : A \vdash f(x) : B(x)}\ ev.$$



How is this evaluation?

Paired with substitution…



Evaluation respects judgemental equality (congruence).

Dependent functions

Computation rule β

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma, x : A \vdash (\lambda y.b(y))(x) \doteq b(x) : B(x)}\ \beta.$$

Computation rule η

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma \vdash \lambda x.\, f(x) \doteq f : \prod_{(x:A)} B(x)}\ \eta.$$

(Dependent) functions

Ordinary functions A -> B are a special case, when codomain has no real dependency.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}}\, W}{\Gamma \vdash \prod_{(x:A)} B \text{ type}}\, \Pi}{\Gamma \vdash A \to B \coloneqq \prod_{(x:A)} B \text{ type}.}$$

defintion

A definition extends the type theory with a new constant and the following rules

$$\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \to B \text{ type}}$$

$$\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \to B \doteq \prod_{(x:A)} B \text{ type}.}$$

Let’s derive a function!

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A\ \text{type}}{\Gamma, x : A \vdash x : A}}{\Gamma \vdash \lambda x.\, x : A \to A}}{\Gamma \vdash \mathsf{id}_A := \lambda x.\, x : A \to A.}$$

$$\mathsf{comp} := \lambda g.\, \lambda f.\, \lambda x.\, g(f(x)).$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash A\ \text{type} \quad \Gamma \vdash B\ \text{type}}{\Gamma, f : B^A, x : A \vdash f(x) : B}\,(a)}{\Gamma, g : C^B, f : B^A, x : A \vdash f(x) : B} \quad \dfrac{\dfrac{\Gamma \vdash B\ \text{type} \quad \Gamma \vdash C\ \text{type}}{\Gamma, g : C^B, y : B \vdash g(y) : C}\,(b)}{\Gamma, g : C^B, f : B^A, y : B \vdash g(y) : C}}{\Gamma, g : C^B, f : B^A, x : A, y : B \vdash g(y) : C}}{\Gamma, g : C^B, f : B^A, x : A \vdash g(f(x)) : C}}{\Gamma, g : C^B, f : B^A \vdash \lambda x.\, g(f(x)) : C^A}}{\Gamma, g : B \to C \vdash \lambda f.\, \lambda x.\, g(f(x)) : B^A \to C^A}}{\Gamma \vdash \lambda g.\, \lambda f.\, \lambda x.\, g(f(x)) : C^B \to (B^A \to C^A)}}{\Gamma \vdash \mathsf{comp} := \lambda g.\, \lambda f.\, \lambda x.\, g(f(x)) : C^B \to (B^A \to C^A).}$$

Composition is associative.

Composition satisfies
left and right unit laws.

Natural numbers

Rules for $\mathbb{N}$:

- formation rule
- introduction rule (zero and successor)
- elimination rule (the induction principle)
- computation rules (how induction principle behaves on zero and successor)
- congruence rules (easy, will not do them)

Natural numbers

Formation rule

$$\frac{}{\vdash \mathbb{N}\ \mathrm{type}}\ \mathbb{N}\text{-form.}$$

Introduction rules

zero element

$$\frac{}{\vdash 0_{\mathbb{N}} : \mathbb{N}}$$

successor function

$$\frac{}{\vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}$$

Natural numbers

Elimination rule

the induction principle

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n)\ \text{type} \\ \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_S : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S) : \prod_{(n:\mathbb{N})} P(n)}\ \mathbb{N}\text{-ind.}$$

Computatuion rules

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n)\ \text{type} \\ \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_S : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, 0_{\mathbb{N}}) \doteq p_0 : P(0_{\mathbb{N}}).}$$

apply to zero

apply to successor

$$\frac{\ldots}{\Gamma, n : \mathbb{N} \vdash \mathsf{ind}_{\mathbb{N}}(p_0, p_S, \mathsf{succ}_{\mathbb{N}}(n)) \doteq p_S(n, \mathsf{ind}_{\mathbb{N}}(p_0, p_S, n)) : P(\mathsf{succ}_{\mathbb{N}}(n)).}$$

Example: defining addition using the induction principle.

$$\mathsf{add}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

$$\mathsf{add}_{\mathbb{N}}(m, 0_{\mathbb{N}}) \doteq m$$

$$\mathsf{add}_{\mathbb{N}}(m, \mathsf{succ}_{\mathbb{N}}(n)) \doteq \mathsf{succ}_{\mathbb{N}}(\mathsf{add}_{\mathbb{N}}(m, n)).$$

by pattern matching
on the second argument

Formally, we want to derive

$$m : \mathbb{N} \vdash \mathsf{add}_{\mathbb{N}}(m) := \mathsf{ind}_{\mathbb{N}}(\mathsf{add\text{-}zero}_{\mathbb{N}}(m), \mathsf{add\text{-}succ}_{\mathbb{N}}(m)) : \mathbb{N} \to \mathbb{N}.$$

where $P(n) := \mathbb{N}$

$$m : \mathbb{N} \vdash \mathsf{add\text{-}zero}_{\mathbb{N}}(m) : \mathbb{N}$$

m

$$m : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(m) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}),$$

$$\mathsf{add\text{-}succ}_{\mathbb{N}}(m, n, x) \doteq \mathsf{succ}_{\mathbb{N}}(x)$$

$$\begin{aligned}\mathsf{add}_{\mathbb{N}}(m, \mathsf{succ}_{\mathbb{N}}(n)) &\doteq \mathsf{ind}_{\mathbb{N}}(\mathsf{add\text{-}zero}_{\mathbb{N}}(m), \mathsf{add\text{-}succ}_{\mathbb{N}}(m), \mathsf{succ}_{\mathbb{N}}(n)) \\ &\doteq \mathsf{add\text{-}succ}_{\mathbb{N}}(m, n, \mathsf{add}_{\mathbb{N}}(m, n)) \\ &\doteq \mathsf{succ}_{\mathbb{N}}(\mathsf{add}_{\mathbb{N}}(m, n)).\end{aligned}$$

$$\dfrac{\dfrac{}{\vdash \mathbb{N}\ \mathrm{type}} \quad \dfrac{\dfrac{}{\vdash \mathbb{N}\ \mathrm{type}} \quad \dfrac{}{\vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{n : \mathbb{N} \vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{m : \mathbb{N}, n : \mathbb{N} \vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}}{m : \mathbb{N} \vdash \lambda n.\, \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}}{m : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(m) :\equiv \lambda n.\, \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N}).}$$

$$\dfrac{\dfrac{\dfrac{\vdots}{m : \mathbb{N} \vdash \mathsf{add\text{-}zero}_{\mathbb{N}}(m) :\equiv m : \mathbb{N}} \quad \dfrac{\vdots}{m : \mathbb{N} \vdash \mathsf{add\text{-}succ}_{\mathbb{N}}(m) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}}{m : \mathbb{N} \vdash \mathsf{ind}_{\mathbb{N}}(\mathsf{add\text{-}zero}_{\mathbb{N}}(m), \mathsf{add\text{-}succ}_{\mathbb{N}}(m)) : \mathbb{N} \to \mathbb{N}}}{m : \mathbb{N} \vdash \mathsf{add}_{\mathbb{N}}(m) :\equiv \mathsf{ind}_{\mathbb{N}}(\mathsf{add\text{-}zero}_{\mathbb{N}}(m), \mathsf{add\text{-}succ}_{\mathbb{N}}(m)) : \mathbb{N} \to \mathbb{N}.}$$

Pattern matching is more **readable**
and **sufficient** for proof assistants.

More inductive types

- Unit type (1)
- Empty type (∅)
- Coproducts (A+B)
- Dependent sum (Σ(x:A)B(x))
- Propositional equality (=)

$$\star : \mathbf{1}$$

$$\mathsf{ind}_{\emptyset} : \prod_{(x:\emptyset)} P(x) \qquad \text{ex-falso} := \mathsf{ind}_{\emptyset} : \emptyset \to A$$

Dependent sum

$$\sum_{(x:A)} B(x)$$

$$\mathsf{pair} : \prod_{(x:A)} \left(B(x) \to \sum_{(y:A)} B(y) \right).$$

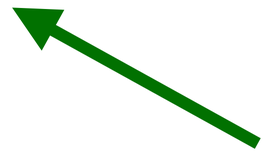
pair 2 (0,0) : Σ (n:$\mathbb{N}$) Vec $\mathbb{N}$ n

$$\mathsf{pr}_1 : \left(\sum_{(x:A)} B(x) \right) \to A$$

$$\mathsf{pr}_2 : \prod_{(p:\sum_{(x:A)} B(x))} B(\mathsf{pr}_1(p)),$$

$$\mathsf{pr}_1(x, y) :\equiv x.$$

$$\mathsf{pr}_2(x, y) :\equiv y.$$



can also be postulated with induction

Specific rules of MLTT

Formation, introduction, elimination, computation and congruence rules for:

- Dependent function type (Π)
- Natural numbers ($\mathbb{N}$)
- Unit type (1)
- Empty type ($\varnothing$)
- Coproducts (A+B)
- Dependent sum (Σ(x:A)B(x))
- Propositional equality (=)

+ definitions we introduce

we will introduce this next

Curry-Howard Correspondence

The Curry-Howard interpretation	
Propositions	Types
Proofs	Elements
Predicates	Type families
$\top$	$\mathbf{1}$
$\bot$	$\emptyset$
$P \vee Q$	$A + B$
$P \wedge Q$	$A \times B$
$P \Rightarrow Q$	$A \to B$
$\neg P$	$A \to \emptyset$
$\exists_x P(x)$	$\sum_{(x:A)} B(x)$
$\forall_x P(x)$	$\prod_{(x:A)} B(x)$
$x = y$	$x = y$

Propositional equality

Formation rule

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b \ \mathsf{type}}$$

Introduction rule (refl)

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}$$

Elimination rule (J-rule)

$$\frac{\Gamma, x : A, y : A, p : x =_A y \vdash P(x, y, p) \ \mathsf{type}}{\Gamma \vdash J : (\Pi_{x:A} P(x, x, \mathsf{refl}_x)) \to \Pi_{a:A} \Pi_{b:A} \Pi_{q:a=_A b} P(a, b, q)}$$

Computation rule

$$\frac{\Gamma, x : A, y : A, p : x =_A y \vdash P(x, y, p) \ \mathsf{type} \qquad \Gamma \vdash d : \Pi_{x:A} P(x, x, \mathsf{refl}_x) \qquad \Gamma \vdash a : A}{\Gamma \vdash J(d, a, a, \mathsf{refl}_a) \equiv d(a) : P(a, a, \mathsf{refl}_a)}$$