Dependant Types

Initiation to research

Louis Milhaud

Université Paris Saclay

January 6, 2024

Outline

1. Introduction
   - History
   - Definition
   - Judgments & Equality
   - Basic rules
2. Type formers
   - $\prod$-type
   - $\sum$-type
   - Naturals
   - Identity types
   - Universes
3. Categoretical Semantics
4. Martin-Lof type theory
   - Properties
   - Intensional and Extensional type theory

History

- 1972: System F, G. Huet
- 1975: Intuitionnistic Type Theory, P. Martin-Lof
- 1986: Nuprl System
- 1988: Calculus of Construction, T. Coquand
- 1989: Coq
- 1997: Syntax & Semantics of Dependant Type Theory, M. Hofmann

Definition

A dependant type is a family of types varying on the elements of another type.

Example:

$$Vec_\sigma(M), \quad M : \mathbb{N}$$

with the following objects:

- $nil_\sigma : Vec_\sigma(0)$
- $Cons_\sigma(U, V) : Vec_\sigma(Succ(M))$

with $U : \sigma$ and $V : Vec_\sigma(M)$

Judgments

$$
\begin{aligned}
&\vdash \Gamma \text{ context} && \Gamma \text{ is a valid context} && (1)\\
\Gamma &\vdash \sigma \text{ type} && \sigma \text{ is a type in context } \Gamma && (2)\\
\Gamma &\vdash M : \sigma && M \text{ is a term of type } \sigma \text{ in context } \Gamma && (3)\\
&\vdash \Gamma = \Delta \text{ context} && \Gamma \text{ and } \Delta \text{ are definitionaly equal contexts} && (4)\\
\Gamma &\vdash \sigma = \tau && \sigma \text{ and } \tau \text{ are def. equal types in context } \Gamma && (5)\\
\Gamma &\vdash M = N : \sigma && M \text{ and } N \text{ are def. equal terms of } \sigma \text{ in } \Gamma && (6)
\end{aligned}
$$

Definitional Equality

Definitional Equality (Per Martin-Lof):

"Definitional equality is intensional equality, or equality of meaning."

This equality defines an equivalence relation:

$$\frac{\vdash \Gamma \text{ context}}{\vdash \Gamma = \Gamma \text{ context}} \qquad \frac{\vdash \Gamma = \Delta \text{ context}}{\vdash \Delta = \Gamma \text{ context}}$$

$$\frac{\vdash \Gamma = \Theta \text{ context} \vdash \Theta = \Delta \text{ context}}{\vdash \Gamma = \Delta \text{ context}}$$

With the same rules for types and terms.

Notions of Equality

Three notions of equality:

- Judgmental equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}} \qquad \frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 = \mathit{Suc}(0) : \mathbb{N}}$$

- Typal equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}} \qquad \frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash \delta_1 : 1 =_{\mathbb{N}} \mathit{Suc}(0)}$$

- Propositional equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}} \qquad \frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 =_{\mathbb{N}} \mathit{Suc}(0) \text{ true}}$$

Basic rules 1

Rules for context formation:

$$\frac{}{\vdash \diamond \text{ context}} \qquad \frac{\Gamma \vdash \sigma \text{ type}}{\vdash \Gamma, x : \sigma \text{ context}}$$

$$\frac{\Gamma = \Delta \text{ context} \quad \Gamma \vdash \sigma = \tau \text{ type}}{\vdash \Gamma, x : \sigma = \Delta, y : \tau \text{ context}}$$

Variable Rule:

$$\frac{\vdash \Gamma, x : \sigma, \Delta \text{ context}}{\Gamma, x : \sigma, \Delta \vdash x : \sigma}$$

Basic rules 2

Rules for typing and definitional equality:

$$\frac{\Gamma \vdash M : \sigma \quad \vdash \Gamma = \Delta \text{ context} \quad \Gamma \vdash \sigma = \tau \text{ type}}{\Delta \vdash M : \tau}$$

$$\frac{\vdash \Gamma = \Delta \text{ context} \quad \Gamma \vdash \sigma \text{ type}}{\Delta \vdash \sigma \text{ type}}$$

Weakening and substitution Rules:

$$\frac{\Gamma, \Delta \vdash \mathcal{J} \quad \Gamma \vdash \rho \text{ type}}{\Gamma, x : \rho, \Delta \vdash \mathcal{J}}$$

$$\frac{\Gamma, x : \rho, \Delta \vdash \mathcal{J} \quad \Gamma \vdash U : \rho}{\Gamma, \Delta[U/x] \vdash \mathcal{J}[U/x]}$$

$\prod$-type

- Called dependant product, dependant function space, π-type...
- Type former for the functions which the return type depends on the element of the entry type.
- Set-theoretic equivalent:
  Cartesian product over a family of sets: $\Pi_{i \in I} B_i$
- type former rules: type formation, term introduction, term elimination, computation rule and an optional uniqueness rule. And type formers are preserved by definitional equality.

Rules

$$\frac{\Gamma \vdash \sigma \text{ type} \quad \Gamma, x : \sigma \vdash \tau \text{ type}}{\Gamma \vdash \Pi x : \sigma.\tau \text{ type}} \textit{Form}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M^{\tau} : \Pi x : \sigma.\tau} \textit{Intro}$$

$$\frac{\Gamma \vdash \lambda x : \sigma.M^{\tau} : \Pi x : \sigma.\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash App_{[x:\sigma]\tau}(\lambda x : \sigma.M^{\tau}, N) = M[N/x] : \tau[N/x]} \textit{Comp}$$

$$\frac{\Gamma \vdash M : \Pi x : \sigma.\tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash App_{[x:\sigma]\tau}(M, N) : \tau[N/x]} \textit{Elim}$$

$\sum$-type

- Type former for pairs which the second element type depends on the first element.
- Set-theoretic equivalent:

  Disjoint union over a family of sets: $\Sigma_{i \in I} B_i \stackrel{\text{def}}{=} \{(i, b) | i \in I \wedge b \in B_i\}$
- $\pi_1 \stackrel{\text{def}}{=} \mathrm{R}^{\Sigma}_{[z:\Sigma x:\sigma.\tau]\rho}([x : \sigma, y : \tau]x, M) : \sigma$
- $\pi_2 \stackrel{\text{def}}{=} \mathrm{R}^{\Sigma}_{[} z : \Sigma x : \sigma.\tau]\tau[z.1/x]([x : \sigma, y : \tau]y, M) : \tau[M.1]$

Rules

$$\frac{\Gamma \vdash \sigma \text{ type} \quad \Gamma, x : \sigma \vdash \tau \text{ type}}{\Gamma \vdash \Sigma x : \sigma.\tau \text{ type}} \mathit{Form}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau[M/x]}{\Gamma \vdash \mathit{Pair}_{[x:\sigma]\tau}(M, N) : \Sigma x : \sigma.\tau} \mathit{Intro}$$

$$\frac{\Gamma \vdash \mathrm{R}^{\Sigma}_{[z:\Sigma x:\sigma.\tau]\rho}([x : \sigma, y : \tau]H, \mathit{Pair}_{[x:\sigma]\tau}(M, N) : \rho[\mathit{Pair}_{[x:\sigma]\tau}(,)/z])}{\Gamma \vdash \mathrm{R}^{\Sigma}_{[z:\Sigma x:\sigma.\tau]\rho}([x : \sigma, y : \tau]H, \mathit{Pair}_{[x:\sigma]\tau}(M, N))} \mathit{Comp}$$

$$= H[M/x, N/y] : \rho[\mathit{Pair}_{[x:\sigma]\tau}(,)/z].$$

$$\Gamma, z : \Sigma x : \sigma.\tau \vdash \rho \text{ type}$$

$$\Gamma, x : \sigma, y : \tau \vdash H : \rho[\mathit{Pair}_{[x:\sigma]\tau}(x, y)/z]$$

$$\frac{\Gamma \vdash M : \Sigma x : \sigma.\tau}{\Gamma \vdash \mathrm{R}^{\Sigma}_{[z:\Sigma x:\sigma.\tau]\rho}([x : \sigma.y : \tau]H, M) : \rho[M/z]} \mathit{Elim}$$

Naturals

Naturals are as always inductively defined, thus we have the two following objects:

$$0 \text{ and } Suc(M) \text{ with } M : \mathbb{N}$$

Then we can define operations like addition:

$$M + N \stackrel{\text{def}}{=} \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(N, [n : \mathbb{N}, x : \mathbb{N}]Suc(x), M) : \mathbb{N}$$

Rules

$$\frac{\vdash \Gamma \text{ context}}{\Gamma \vdash \mathbb{N} \text{ type}} \mathit{Form} \quad \frac{\vdash \Gamma \text{ context}}{\Gamma \vdash 0 : \mathbb{N}} \,\&\, \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \mathit{Suc}(M) : \mathbb{N}} \mathit{Intro}$$

$$\frac{\begin{array}{c}\Gamma, n : \mathbb{N} \vdash \sigma \text{ type} \\ \Gamma \vdash H_z : \sigma[0/n] \\ \Gamma, n : \mathbb{N}, x : \sigma \vdash H_s : \sigma[\mathit{Suc}(n)/n] \\ \Gamma \vdash M : \mathbb{N}\end{array}}{\Gamma \vdash \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, M) : \sigma[M/n]} \mathit{Elim}$$

$$\frac{\Gamma \vdash \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, 0) : \sigma[0/n]}{\Gamma \vdash \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, 0) = H_z : \sigma[0/n]} \mathit{Comp}$$

$$\frac{\Gamma \vdash \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, \mathit{Suc}(M)) : \sigma[\mathit{Suc}(M)/n]}{\Gamma \vdash \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, \mathit{Suc}(M)) =} \mathit{Comp}$$

$$H_s[M/n, \mathrm{R}^{\mathbb{N}}_{[n:\mathbb{N}]\sigma}(H_z, [n : \mathbb{N}, x : \sigma]H_s, M)/x] : \sigma[\mathit{Suc}(M)/n]$$

Identity types

- Type former for typal equality
- Enables equality reasonning inside type theory
- $Refl_\sigma(M) : Id_\sigma(M, M)$ with $M : \sigma$
- We can deduce other properties of typal equality like symmetry, transitivity, Leibniz principle thanks to the elimination rule.

Rules

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash Id_\sigma(M, N) \text{ type}} \, Form$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash Refl_\sigma(M) : Id_\sigma(M, M)} \, Intro$$

$$\frac{\begin{array}{c}\Gamma \vdash \sigma \text{ type} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma \\ \Gamma, x : \sigma, y : \sigma, p : Id_\sigma(x, y) \vdash \tau \text{ type} \\ \Gamma, z : \sigma \vdash H : \tau[z/x, z/y, Refl_\sigma(z)/p] \\ \Gamma \vdash P : Id_\sigma(M, N)\end{array}}{\Gamma \vdash \mathrm{R}^{Id}_{[x:\sigma, y:\sigma, p:Id_\sigma(x,y)]\tau}([z:\sigma]H, M, N, P) : \tau[M/x, N/y, P/p]} \, Elim$$

$$\frac{\Gamma \vdash \mathrm{R}^{Id}_{[x:\sigma, y:\sigma, p:Id_\sigma(x,y)]\tau}([z:\sigma]H, M, M, Refl_\sigma(M)) : \tau[M/x, M/y, Refl_\sigma(M)/p]}{\Gamma \vdash \ idem = H[M/z] : \tau[M/x, M/y, Refl_\sigma(M)/p]}$$

Comp

Universes

Rules

$$\frac{\vdash \Gamma \text{ context}}{\Gamma \vdash U \text{ type}} \qquad \frac{\Gamma \vdash M : U}{\Gamma \vdash El(M) \text{ type}}$$

Categoretical Semantics: the idea

- Let A type interprets as an object in some category $\mathcal{C}$
- Let $x : A \vdash B$ type interprets as a morphism $B \to A$ in $\mathcal{C}$
- Let $x : A \vdash B$ type interprets as an object in the slice category $\mathcal{C}_{/A}$
- Let interprets as a pullback But...

Comprehension Category

Definition:

A comprehension category consists of a strictly commutative triangle of functors:

$$\begin{tikzcd} E \arrow[r] \arrow[dr] & C^I \arrow[d, "\mathrm{cod}"] \\ & C \end{tikzcd}$$

where C I is the arrow category of C and cod:C I→C denotes the codomain projection (which is a fibration if C has pullbacks), and such that
E→C is a Grothendieck fibration,
E→C I takes cartesian morphisms in E to cartesian morphisms in C I (i.e. to pullback squares in C).

Category with families

A category with attributes specifies for each "context" only a set of "types" in that context. A comprehension category, by contrast, specifies a whole category of "types" in each context. If $A, B \in E\Gamma$, then we may think of a morphism $f : E \to A$ in $E\Gamma$ as a term in type theory. A category with families specifies instead, for each context and each type in that context, a set of "terms belonging to that type". These should be thought of as terms in type theory.

Martin-Lof type theory: Axiom of choice

Logical form of the axiom of choice:

Let σ, τ, υ three types such that:

$$\frac{}{\vdash \sigma \text{ type}} \quad \frac{\vdash x : \sigma}{x : \sigma \vdash \tau \text{ type}} \quad \frac{\vdash x : \sigma \quad \vdash y : \tau}{x : \sigma, y : \tau \vdash \upsilon \text{ type}}$$

Axiom of choice:

$$(\forall x : \sigma)(\exists y : \tau)\upsilon \to (\exists f : (\Pi x : \sigma)\tau)(\forall x : \sigma)\upsilon$$

Intensional and Extensional type theory

- type theory $+$ identity types $=$ intensional type theory
- equality reflection:

$$\frac{\Gamma \vdash M = N : \sigma}{\Gamma \vdash P : Id_\sigma(M, N)}$$

- type theory $+$ identity types $+$ reflection rule $=$ extensional type theory
- Judgmental equality and typing undecidable