

Dependant Types

Initiation to research

Louis Milhaud

Université Paris Saclay

January 8, 2024

Outline

1 Introduction

- History
- Definition
- Judgments & Equality
- Basic rules

2 Type formers

- Π -type
- Σ -type
- Naturals
- Identity types
- Universes

3 Intensional and Extensional type theory

4 Categorical Semantics

History

- 1972: System F, J-Y. Girard
- 1975: Intuitionistic Type Theory, P. Martin-Lof
- 1986: Nuprl System
- 1988: Calculus of Construction, T. Coquand
- 1989: Coq
- 1997: Syntax & Semantics of Dependent 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

$\vdash \Gamma$ context

Γ is a valid context (1)

$\Gamma \vdash \sigma$ type

σ is a type in context Γ (2)

$\Gamma \vdash M : \sigma$

M is a term of type σ in context Γ (3)

$\vdash \Gamma = \Delta$ context

Γ and Δ are definitionally equal contexts (4)

$\Gamma \vdash \sigma = \tau$

σ and τ are def. equal types in context Γ (5)

$\Gamma \vdash M = N : \sigma$

M and N are def. equal terms of σ in Γ (6)

Definitional Equality

Definitional Equality (Per Martin-Lof):

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

This equality defines an equivalence relation:

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

With the same rules for types and terms.

Notions of Equality

Three notions of equality:

- Judgmental equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}}$$

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

- Typal equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}}$$

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

- Propositional equality

$$\frac{\Gamma \vdash \mathbb{N} \text{ type}}{\Gamma \vdash 1 : \mathbb{N}}$$

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

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:

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

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

Π -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: $\prod_{i \in I} B_i$
- type former rules: type formation, term introduction, term elimination, computation rule and an optional uniqueness rule. And type formers are preserved by definitional equality.

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

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

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

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

Σ -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: $\sum_{i \in I} B_i \stackrel{\text{def}}{=} \{(i, b) | i \in I \wedge b \in B_i\}$

- $\pi_1 \stackrel{\text{def}}{=} R_{[z : \sum x : \sigma. \tau] \rho}^{\Sigma}([x : \sigma, y : \tau]x, M) : \sigma$
- $\pi_2 \stackrel{\text{def}}{=} R_{[z : \sum x : \sigma. \tau] \tau[z.1/x]}^{\Sigma}([x : \sigma, y : \tau]y, M) : \tau[M.1]$

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

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

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

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

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

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

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

Naturals

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

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

Then we can define operations like addition:

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

$$\begin{array}{c}
 \frac{\vdash \Gamma \text{ context}}{\Gamma \vdash \mathbb{N} \text{ type}} \textit{Form} \quad \frac{\vdash \Gamma \text{ context}}{\Gamma \vdash 0 : \mathbb{N}} \& \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \text{Suc}(M) : \mathbb{N}} \textit{Intro} \\
 \Gamma, n : \mathbb{N} \vdash \sigma \text{ type} \\
 \Gamma \vdash H_z : \sigma[0/n] \\
 \Gamma, n : \mathbb{N}, x : \sigma \vdash H_s : \sigma[\text{Suc}(n)/n] \\
 \Gamma \vdash M : \mathbb{N} \\
 \hline
 \Gamma \vdash R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, M) : \sigma[M/n] \textit{Elim} \\
 \Gamma \vdash R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, 0) : \sigma[0/n] \\
 \hline
 \Gamma \vdash R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, 0) = H_z : \sigma[0/n] \textit{Comp} \\
 \Gamma \vdash R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, \text{Suc}(M)) : \sigma[\text{Suc}(M)/n] \\
 \hline
 \Gamma \vdash R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, \text{Suc}(M)) = \\
 H_s[M/n, R_{[n:\mathbb{N}]\sigma}^{\mathbb{N}}(H_z, [n : \mathbb{N}, x : \sigma]H_s, M)/x] : \sigma[\text{Suc}(M)/n] \textit{Comp}
 \end{array}$$

Identity types

- Type former for typal equality
- Enables equality reasoning inside type theory
- $\text{Refl}_\sigma(M) : \text{Id}_\sigma(M, M)$ with $M : \sigma$
- We can deduce other properties of typal equality like symmetry, transitivity, Leibniz principle thanks to the elimination rule.

Rules

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

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

$$\Gamma \vdash \sigma \text{ type} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma$$

$$\Gamma, x : \sigma, y : \sigma, p : Id_{\sigma}(x, y) \vdash \tau \text{ type}$$

$$\Gamma, z : \sigma \vdash H : \tau[z/x, z/y, Refl_{\sigma}(z)/p]$$

$$\frac{\Gamma \vdash P : Id_{\sigma}(M, N)}{\Gamma \vdash R_{[x:\sigma, y:\sigma, p:Id_{\sigma}(x,y)]\tau}^{Id}([z:\sigma]H, M, N, P) : \tau[M/x, N/y, P/p]} \text{Elim}$$

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

Comp

Universes

- Universes are type formers containing codes for types
- We have the two basic rules:

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

- M is a code of the universe U
- $El(M)$ is the type associated to this code
- Add type formers to our Universes
- Rules to maintain closure

Adding dependant type $\hat{\Pi}$

$$\frac{\Gamma \vdash S : U \quad \Gamma, s : El(S) \vdash T : U}{\Gamma \vdash \hat{\Pi}(S, [s : El(S)] T) : U} \text{Form}$$

$$\frac{\Gamma \vdash \hat{\Pi}(S, [s : El(S)] T) : U}{\Gamma \vdash El(\hat{\Pi}(S, [s : El(S)] T)) = \Pi s : El(S). El(T) \text{ type}}$$

Impredicative quantification

- We can add \forall or Impredicative quantification
- Terms like $\forall x : \sigma. T : U$
- domain type σ *arbitrarily big*
- Allows typing of the identity function *polyone* known from polymorphic lambda calculus
- $\text{polyone} = \forall c : U. \forall : El(c). c$

Impredicative quantification - Rules

$$\frac{\Gamma \vdash \sigma \text{ type} \quad \Gamma, x : \sigma \vdash T : U}{\Gamma \vdash \forall x : \sigma. T : U}$$
$$\frac{\Gamma, x : \sigma \vdash M : El(T)}{\Gamma \vdash \hat{\lambda}x : \sigma. M^{El(T)} : El(\forall x : \sigma. T)} \text{Intro}$$
$$\frac{\Gamma \vdash M : el(\forall x : \sigma. T) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(M, N) : El(T[N/x])} \text{Elim}$$
$$\frac{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(\hat{\lambda}x : \sigma. M^{El(T)}, N) : El(T[N/x])}{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(\hat{\lambda}x : \sigma. M^{El(T)}, N) = M[N/x] : El(T[N/x])} \text{Comp}$$

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

Context morphisms

Let Γ and $\Delta \stackrel{\text{def}}{=} x_1 : \sigma_1, \dots, x_n : \sigma_n$ be valid contexts.

$f \stackrel{\text{def}}{=} (M_1, \dots, M_n)$ is a context morphism (we write $\Gamma \vdash f \implies \Delta$)
when the following n judgements hold:

$$\Gamma \vdash M_1 : \sigma_1$$

$$\Gamma \vdash M_2 : \sigma_2[M_1/x_1]$$

...

$$\Gamma \vdash M_n : \sigma_n[M_1/x_1][M_2/x_2] \dots [M_{n-1}/x_{n-1}]$$

Categorical Semantics

Our basic Syntax contains now 4 objects

- contexts
- contexts morphisms
- types in Ty
- terms in Tm

And we define the corresponding Semantic:

- \mathcal{C} a category of contexts and context morphisms
- for $\Gamma \in \mathcal{C}$ a collection $Ty_{\mathcal{C}}(\Gamma)$ of semantic types
- for $\Gamma \in \mathcal{C}$ and $\sigma \in Ty_{\mathcal{C}}$ a collection $Tm_{\mathcal{C}}(\Gamma, \sigma)$ of semantic terms

Category \mathcal{Fam}

The category \mathcal{Fam} of families of sets has:

- as objects pairs $A = (A^0, A^1)$
- as arrows f between A and B a pair (f^0, f^1)

We also define the functor $\mathcal{F} : \mathcal{C}^{op} \rightarrow \mathcal{Fam}$ such that:

$$\mathcal{F}(\Gamma) = (Ty_{\mathcal{C}}(\Gamma), (Tm_{\mathcal{C}}(\Gamma, \sigma))_{\sigma \in Ty_{\mathcal{C}}(\Gamma)})$$

Category with families

We can now define a category with families with:

- a category \mathcal{C} with terminal object
- a functor $\mathcal{F} = (Ty, Tm) : \mathcal{C}^{op} \rightarrow \mathcal{Fam}$
- a comprehension for each $\Gamma \in \mathcal{C}$ and $\sigma \in Ty_{\mathcal{C}}(\Gamma)$