Dependant Types

Initiation to research

Louis Milhaud

Université Paris Saclay

January 7, 2024

Outline

History

- 1972: System F, J-Y. Girard
- 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

$\vdash \Gamma \text{ context}$	Γ is a valid context	(1)
$\Gamma \vdash \sigma \text{ type}$	σ is a type in context Γ	(2)
$\Gamma \vdash M : \sigma$	M is a term of type σ in context Γ	(3)
$\vdash \Gamma = \Delta \text{ context}$	Γ and Δ are definitionaly 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:

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

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

With the same rules for types and terms.

Notions of Equality

Three notions of equality:

- Judgmental equality

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

- 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

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

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

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

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

$\sum$-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: $\Sigma_{i \in I} B_i \overset{\text{def}}{=} \{(i, b) | i \in I \wedge b \in B_i\}$
- $\pi_1 \overset{\text{def}}{=} \mathrm{R}^{\Sigma}_{[z:\Sigma x:\sigma.\tau]\rho}([x : \sigma, y : \tau]x, M) : \sigma$
- $\pi_2 \overset{\text{def}}{=} \mathrm{R}^{\Sigma}_{[}z : \Sigma x : \sigma.\tau]\tau[z.1/x]([x : \sigma, y : \tau]y, M) : \tau[M.1]$

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:

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

Rules

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

$$\Gamma, n : \mathbb{N} \vdash \sigma \text{ type}$$

$$\Gamma \vdash H_z : \sigma[0/n]$$

$$\Gamma, n : \mathbb{N}, x : \sigma \vdash H_s : \sigma[\mathit{Suc}(n)/n]$$

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

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

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

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

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

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

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

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

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

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

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

Comp

Universes

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

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

- 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 $\hat{\Pi}$

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

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

Impredicative quantification

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

Impredicative quantification - Rules

$$\frac{\Gamma \vdash \sigma \text{ type} \quad \Gamma, x : \sigma \vdash T : U}{\Gamma \vdash \forall x : \sigma . T : U}$$

$$\frac{\Gamma, x : \sigma \vdash M : El(T)}{\Gamma \vdash \hat{\lambda} x : \sigma . M^{El(T)} : El(\forall x : \sigma . T)} Intro$$

$$\frac{\Gamma \vdash M : el(\forall x : \sigma . T) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(M, N) : El(T[N/x])} Elim$$

$$\frac{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(\hat{\lambda} x : \sigma . M^{El(T)}, N) : El(T[N/x])}{\Gamma \vdash \hat{App}_{[x:\sigma]El(T)}(\hat{\lambda} x : \sigma . M^{El(T)}, N) = M[N/x] : El(T[N/x])} Comp$$

Intensional and Extensional type theory

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

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

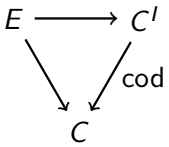
- type theory + identity types + reflection rule = extensional type theory
- Judgmental equality and typing undecidable

Categoretical Semantics: the idea

- Let $A\ \mathrm{type}$ interprets as an object in some category $\mathcal{C}$
- Let $x : A \vdash B\ \mathrm{type}$ interprets as a morphism $B \to A$ in $\mathcal{C}$
- Let $x : A \vdash B\ \mathrm{type}$ 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:



where C I is the arrow category of C and cod:C I$\to$C denotes the codomain projection (which is a fibration if C has pullbacks), and such that
E$\to$C is a Grothendieck fibration,
E$\to$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.