# Lectures on Dependent Type Theory

Nicola Gambino

March 15th, 2009

**Abstract**

We give a short overview of the material that will be presented during the lectures on Dependent Type Theory to be given at the Midlands Graduate School in the Foundations of Computing 2009.

## 1 Dependent type theories

The first lecture will be devoted to introducing dependent type theories. We consider dependent type theories with the following four kinds of judgements:

$$A \in \mathrm{Type}\,, \quad A = B \in \mathrm{Type}\,, \quad a \in A\,, \quad a = b \in A\,.$$

These judgements express that $A$ is a type, that $A$ and $B$ are definitionally equal types, that the term $a$ is an element of the type $A$, and that the terms $a$ and $b$ are definitionally equal elements of the type $A$, respectively. Of course, we allow such judgements to be made relative to a context $\Gamma$ of variable declarations of the form

$$\Gamma = (x_0 : A_0, x_1 : A_1, \ldots, x_n : A_n)\,.$$

For each form of type we give four kinds of rules:

- formation rules,
- introduction rules,
- elimination rules,
- computation rules.

The rules for the most of the forms of types that we consider during the lectures are given in Appendix A. The lecture will explain the general scheme that motivates the formulation of these rules and the different roles that are played by the four kinds of rules. To illustrate in practice how the rules for dependent type theories interact between each other, we will consider some derived rules.

## 2 Propositions-as-types

The second lecture will be devoted to studying the propositions-as-types idea in the context of dependent type theories. To make this idea precise, we will consider extensions of the dependent type theories considered earlier with the following forms of judgement

$$\phi \in \mathrm{Prop}\,, \qquad \phi_1, \ldots \phi_n \vdash \phi\,.$$

These jdugements express that $\phi$ is a proposition and that a proposition $\phi$ follows logically from the assumptions $\phi_1, \ldots \phi_n$. We refer to such dependent type theories as *logic-enriched dependent type theories*. The rules for a logic-enriched dependent type theory can be seen as those of a many-sorted logic in which the sorts are given by the types of the underlying dependent type theory. The propositions-as-types idea can then be formulated as a syntactic translation from certain logic-enriched type theories into their underlying dependent type theories, defined as follows:

$$\begin{aligned}
[\![\bot]\!] &= 0\,,\\
[\![\top]\!] &= 1\,,\\
[\![\phi \wedge \psi]\!] &= [\![\phi]\!] \times [\![\psi]\!]\,,\\
[\![\phi \vee \psi]\!] &= [\![\phi]\!] + [\![\psi]\!]\,,\\
[\![\phi \supset \psi]\!] &= [\![\phi]\!] \to [\![\psi]\!]\,,\\
[\![(\forall x : A)\phi]\!] &= (\Pi x : A)[\![\phi]\!]\,,\\
[\![(\exists x : A)\phi]\!] &= (\Sigma x : A)[\![\phi]\!]\,.
\end{aligned}$$

We will show that the propositions-as-types translation validates not only the standard rules for intuitionistic first-order logic, but also induction rules for all the inductive types of the underlying dependent type theory, and the so-called type-theoretic axiom of choice. This will lead to a characterisation of the propositions that are valid under the proposition-as-types translation.

## 3 Semantics in locally cartesian closed categories

The third and fourth lecture will be devoted to the semantics of dependent type theories. To begin with, we consider semantics in locally cartesian closed categories. One of the distinguishing features of locally cartesian categories is that, for every map $f : B \to A$ in a locally cartesian closed category $\mathbb{C}$, the pullback functor $\Delta_f : \mathbb{C}/A \to \mathbb{C}/B$ has both a left adjoint $\Sigma_f : \mathbb{C}/B \to \mathbb{C}/A$ and a right adjoint $\Pi_f : \mathbb{C}/B \to \mathbb{C}/A$. We will describe the analogy between these adjoints and $\Sigma$-types and $\Pi$-types. We then discuss how the idea of interpreting dependent type theories in locally cartesian closed suffers from two distinct problems. The first problem is an issue of completeness: the semantics in locally cartesian closed categories validates more rules than the ones we assumed as part of the dependent type theories. The second problem concerns coherence issues: while substitution satisfies strictly both an associative law and commutation law with type-constructors, its semantic counterpart does not. To address the first problem, we discuss the effect of extending our dependent type theories with rules that are valid in any locally cartesian closed categories, thus arriving at the formulation of *extensional dependent type theories*. To address the second problem, we revisit the semantics of dependent type theories in the general context of the theory of fibrations, where a 'strictification result' allows to solve coherence issues for the semantics of extensional dependent type theories.

## 4 Homotopical aspects of dependent type theories

The fourth lecture will be devoted to illustrate some recent research aimed at developing a satisfactory semantics for dependent type theories which are not extensional. This research involves surprising new connections with homotopical algebra and higher-dimensional category theory. First, we will show how categories equipped with a weak factorisation system satisfying appropriate coherence conditions provide a semantics for identity types. Secondly, we will show how the very syntax of dependent type theories with rules for identity types gives rise to a category equipped with a weak factorisation system.

# A Deduction rules for dependent type theories

$$\frac{}{0 \in \mathrm{Type}}$$

$$\frac{e \in 0 \qquad (x \in 0)\ C(x) \in \mathrm{Type}}{\mathrm{rec}(e) \in C(e)}$$

Table 1: Rules for the empty type

$$\frac{}{1 \in \mathrm{Type}}$$

$$\frac{}{* \in 1}$$

$$\frac{e \in 1 \qquad (x \in 1)\ C(x) \in \mathrm{Type} \qquad c \in C(*)}{\mathrm{rec}(e, c) \in C(e)}$$

$$\frac{(x \in 1)\ C(x) \in \mathrm{Type} \qquad c \in C(*)}{\mathrm{rec}(*, c) = c \in C(*)}$$

Table 2: Rules for the one-element type

$$\frac{A \in \mathrm{Type} \qquad B \in \mathrm{Type}}{A + B \in \mathrm{Type}}$$

$$\frac{a \in A}{\iota_A(a) \in A + B}$$

$$\frac{b \in B}{\iota_B(b) \in A + B}$$

$$\frac{e \in A + B \quad (z \in A + B)\ C(z) \in \mathrm{Type} \quad (x \in A)\ c(x) \in C(\iota_A(x)) \quad (y \in B)\ d(y) \in C(\iota_B(y))}{\mathrm{rec}(e, c, d) \in C(e)}$$

$$\frac{a \in A \quad (z \in A + B)\ C(z) \in \mathrm{Type} \quad (x \in A)\ c(x) \in C(\iota_A(x)) \quad (y \in B)\ d(y) \in C(\iota_B(y))}{\mathrm{rec}(\iota_A(a), c, d) = c(a) \in C(\iota_A(a))}$$

$$\frac{b \in B \quad (z \in A + B)\ C(z) \in \mathrm{Type} \quad (x \in A)\ c(x) \in C(\iota_A(x)) \quad (y \in B)\ d(y) \in C(\iota_B(y))}{\mathrm{rec}(\iota_B(b), c, d) = d(b) \in C(\iota_B(b))}$$

Table 3: Deduction rules for disjoint union of types

$$\frac{}{\text{Nat} \in \text{Type}}$$

$$\frac{}{0 \in \text{Nat}}$$

$$\frac{n \in \text{Nat}}{\text{succ}(n) \in \text{Nat}}$$

$$\frac{n \in \text{Nat} \qquad (x \in \text{Nat})\ C(x) \in \text{Type} \qquad c \in C(0) \qquad (x \in \text{Nat}, y \in C(x))\ d(x,y) \in C(\text{succ}(x))}{\text{natrec}(n,c,d) \in C(n)}$$

$$\frac{(x \in \text{Nat})\ C(x) \in \text{Type} \qquad c \in C(0) \qquad (x \in \text{Nat}, y \in C(x))\ d(x,y) \in C(\text{succ}(x))}{\text{natrec}(0,c,d) = c \in C(0)}$$

$$\frac{n \in \text{Nat} \qquad (x \in \text{Nat})\ C(x) \in \text{Type} \qquad c \in C(0) \qquad (x \in \text{Nat}, y \in C(x))\ d(x,y) \in C(\text{succ}(x))}{\text{natrec}(\text{succ}(n),c,d) = d(n, \text{natrec}(n,c,d)) \in C(\text{succ}(n))}$$

Table 4: Deduction rules for the type of natural numbers

$$\frac{(x \in A)\ B(x) \in \text{Type}}{(\Pi x \in A)B(x) \in \text{Type}}$$

$$\frac{(x \in A)\ f(x) \in B(x)}{(\lambda x \in A)f(x) \in (\Pi x \in A)B(x)}$$

$$\frac{f \in (\Pi x \in A)B(x) \qquad a \in A}{\text{app}(f,a) \in B(a)}$$

$$\frac{(x \in A)\ f(x) \in B(x) \qquad a \in A}{\text{app}((\lambda x \in A)f(x), a) = f(a) \in B(a)}$$

Table 5: Deduction rules for Π-types

$$\frac{(x \in A) \quad B(x) \in \mathrm{Type}}{(\Sigma x \in A)B(x) \in \mathrm{Type}}$$

$$\frac{a \in A \qquad b \in B(a)}{\mathrm{pair}(a,b) \in (\Sigma x \in A)B(x)}$$

$$\frac{c \in (\Sigma x \in A)B(x) \qquad (z \in (\Sigma x \in A)B(x))\ C(z) \in \mathrm{Type} \qquad (x \in A, y \in B(x))\ d(x,y) \in C(\mathrm{pair}(x,y))}{\mathrm{split}(c,d) \in C(c)}$$

$$\frac{a \in A \qquad b \in B(a) \qquad (z \in (\Sigma x \in A)B(x))\ C(z) \in \mathrm{Type} \qquad (x \in A, y \in B(x))\ d(x,y) \in C(\mathrm{pair}(x,y))}{\mathrm{split}(\mathrm{pair}(a,b),d) = d(a,b) \in C(\mathrm{pair}(a,b))}$$

Table 6: Deduction rules for $\Sigma$-types

$$\frac{A \in \mathrm{Type} \quad a \in A \quad b \in A}{\mathrm{Id}_A(a,b) \in \mathrm{Type}}$$

$$\frac{a \in A}{\mathrm{r}(a) \in \mathrm{Id}_A(a,a)}$$

$$\frac{p \in \mathrm{Id}_A(a,b) \qquad (x \in A, y \in A, z \in \mathrm{Id}_A(x,y))\ C(x,y,z) \in \mathrm{Type} \qquad (x \in A)\ d(x) \in C(x,x,\mathrm{r}(x))}{\mathrm{J}(a,b,p,d) \in C(a,b,p)}$$

$$\frac{a \in A \qquad (x \in A, y \in A, z \in \mathrm{Id}_A(x,y)) \quad C(x,y,z) \in \mathrm{Type} \qquad (x \in A) \quad d(x) \in C(x,x,\mathrm{r}(x))}{\mathrm{J}(a,a,\mathrm{r}(a),d) = d(a) \in C(a,a,\mathrm{r}(a))}$$

Table 7: Deduction rules for identity types