Dependent Type Theory

Lecture 1

Nicola Gambino

MGS - Leicester 2009

References

B. Nordstöm, K. Petersson, and J. M. Smith
Martin-Löf's Type Theory
Handbook of Logic in Computer Science, Vol. 5
Oxford University Press, 2001.

B. Nordström, K. Petersson, and J. M. Smith
Programming in Martin-Löf's Type Theory
Oxford University Press, 1990.

Outline

Categorical judgements

There are four forms of categorical judgements:

- $A \in \mathrm{Type}$
- $A = B \in \mathrm{Type}$
- $a \in A$
- $a = b \in A$

These express, respectively, that:

- A is a type
- A and B are definitionally equal types
- a is an element of A
- a and b are definitionally equal elements of A.

Hypothetical judgements (I)

There are four forms of hypothetical judgements:

- $(\Gamma)\ A \in \mathrm{Type}$
- $(\Gamma)\ A = B \in \mathrm{Type}$
- $(\Gamma)\ a \in A$
- $(\Gamma)\ a = b \in A$.

Here Γ is a **context** of variable declarations of the form

$$(\Gamma) = \left(x_0 \in A_0, x_1 \in A_1(x_0), \ldots, x_n \in A_n(x_0, \ldots, x_{n-1})\right).$$

Hypothetical judgements (II)

For the context Γ to be well-formed we need to know that the judgements

$$A_0 \in \mathrm{Type}$$
$$(x_0 \in A_0)\ A_1(x_0) \in \mathrm{Type}$$
$$\vdots$$
$$\big(x_0 \in A_0, \ldots, x_{n-1} \in A_{n-1}(x_0, \ldots, x_{n-2})\big)\ A_n(x_0, \ldots, x_{n-1}) \in \mathrm{Type}$$

are derivable.

Deduction rules

Deduction rules have the form

$$\frac{(\Gamma_1)\ J_1 \qquad \cdots \qquad (\Gamma_n)\ J_n}{(\Gamma)\ J}$$

We often omit contexts that are common to premisses and conclusion.

Outline

1. Preliminaries $\surd$
2. The dependent type theory **ML**

The dependent type theory **ML** (I)

Two groups of rules:

- General deduction rules, *e.g.*

$$\frac{(\Gamma)\ A \in \mathrm{Type}}{(\Gamma, x \in A)\ x \in A}$$

- Deduction rules for the following forms of type:

$$0\,, \quad 1\,, \quad \mathrm{Nat}\,, \quad A + B\,,$$

$$\mathrm{Id}_A(a, b)\,, \quad (\Sigma x \in A)B(x)\,, \quad (\Pi x \in A)B(x)\,.$$

The dependent type theory **ML** (II)

For each form of type, we give four groups of deduction rules:

- Formation rules
- Introduction rules
- Elimination rules
- Computation rules

Deduction rules for the type of natural numbers (I)

Formation rule:

$$\mathsf{Nat} \in \mathsf{Type}$$

Introduction rules:

$$0 \in \mathsf{Nat} \qquad \frac{n \in \mathsf{Nat}}{\mathsf{succ}(n) \in \mathsf{Nat}}$$

Deduction rules for the type of natural numbers (II)

Elimination rule:

$$\frac{c \in \mathrm{Nat} \qquad d \in C(0) \qquad (x \in \mathrm{Nat}, y \in C(x))\ e(x,y) \in C(\mathrm{succ}(x))}{\mathrm{natrec}(c,d,e) \in C(c)}$$

Computation rules:

$$\frac{d \in C(0) \qquad (x \in \mathrm{Nat}, y \in C(x))\ e(x,y) \in C(\mathrm{succ}(x))}{\mathrm{natrec}(0,d,e) = d \in C(0)}$$

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

Deduction rules for the unit type (I)

Formation rules:

$$1 \in \mathsf{Type}$$

Introduction rule:

$$* \in 1$$

Deduction rules for the unit type (II)

Elimination rule:

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

Computation rule:

$$\frac{d \in C(*)}{\mathrm{rec}(*, d) = d \in C(*)}$$

Deduction rules for the empty type (I)

Formation rule:

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

Introduction rules:

Deduction rules for the empty type (II)

Elimination rule:

$$\frac{c \in 0}{\mathrm{rec}(c) \in C(c)}$$

Computation rules:

Deduction rules for sum types (I)

Formation rules:

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

Introduction rules:

$$\frac{a \in A}{\mathrm{inl}(a) \in A + B} \qquad \frac{b \in B}{\mathrm{inr}(b) \in A + B}$$

Deduction rules for sum types (II)

Elimination rule:

$$\frac{c \in A + B \quad (x \in A)\ d(x) \in C(\mathrm{inl}(x)) \quad (y \in B)\ e(y) \in C(\mathrm{inr}(y))}{\mathrm{case}(c, d, e) \in C(c)}$$

Computation rules:

$$\frac{a \in A \quad (x \in A)\ d(x) \in C(\mathrm{inl}(x)) \quad (y \in B)\ e(y) \in C(\mathrm{inr}(y))}{\mathrm{case}(\mathrm{inl}(a), d, e) = d(a) \in C(\mathrm{inl}(a))}$$

$$\frac{b \in B \quad (x \in A)\ d(x) \in C(\mathrm{inl}(x)) \quad (y \in B)\ e(y) \in C(\mathrm{inr}(y))}{\mathrm{case}(\mathrm{inr}(b), d, e) = e(b) \in C(\mathrm{inr}(b))}$$

Deduction rules for identity types (I)

Formation rule:

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

Introduction rule:

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

Deduction rules for identity types (II)

Elimination rule:

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

Computation rule:

$$\frac{a \in A \qquad (x \in A)\ d(x) \in C(x,x,\mathrm{r}(x))}{\mathrm{J}(a,a,\mathrm{r}(a),d) = d(a) \in C(a,a,\mathrm{r}(a))}$$

Deduction rules for Σ-types (I)

Formation rule:

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

Introduction rule:

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

Deduction rules for Σ-types (II)

Elimination rule:

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

Computation rule:

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

Deduction rules for Π-types (I)

Formation rule:

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

Introduction rule:

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

Deduction rules for Π-types (II)

Elimination rule:

$$\frac{b \in (\Pi x \in A)B(x) \in \mathrm{Type} \qquad a \in A}{\mathrm{app}(b, a) \in B(a)}$$

Computation rule:

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

Remarks

- We used ‘Type’ for what is called ‘Set’ in the Handbook paper.
- We did not use explicitly any logical framework.