Martin Löf's Type Theory: constructive set theory and logical framework

Marco Gaboardi

Dipartimento di Informatica - Università degli Studi di Torino

Brief History

- (Impredicative) Type Theory.

  1971 Per Martin-Löf, A theory of Types.

- (Predicative) Type Theory as Constructive Set Theory.

  1979 Per Martin-Löf, Constructive Mathematics and Computer Programming .

  1984 Per Martin-Löf, Intuitionistic Type Theory.

- (Predicative) Type Theory as Logical Framework

  1985 Per Martin-Löf, Truth of a Proposition, Evidence of a Judgement, Validity of a Proof .

  1990 Bengt Nordström, Kent Petersson and Jan M. Smith, Programming in Martin-Löf's Type Theory: an introduction.

Ideas

- The main idea behind Martin-Löf’s type theory is Curry-Howard Isomorphism:
  - correspondence between propositions and sets (type theory as constructive set theory)
  - correspondence between proofs and programs (type theory as programming language
- One of Martin-Löf’s original aims with type theory was that it could serve as a framework in which other theories could be interpreted, inheriting interesting properties: normalization, etc.
- Semantics in Martin-Löf’s type theory is not given “ a posteriori”, instead it is independent and come before as a justification of the formal system . The meaning of type theory is explained in terms of computations.

Computations - Informally

The meaning of type theory is explained in terms of computations. Hence computation is a primitive concept. We can regard computation as the process of evaluation of terms:

$$T \Downarrow T'$$

It is important to distinguish beetween input data (elements) and output value ((lazy) canonical elements) canonical elements. Usually we have evaluation of terms T to values V:

$$T \Downarrow V$$

Usually we consider computations of a term T as composition of evaluation on subterms:

1. evaluating the subterm T_1 we obtain a canonical element V_1
2. operating on T by means of V_1 we obtain T_2
3. evaluating T_2 we obtain a canonical element V

$$\frac{T_1 \Downarrow V_1 \quad T_2 \Downarrow V}{T \Downarrow V}$$

The meaning of type theory is explained in terms of “operational” processes.

Proposition and Judgement

- It is essential in Martin-Löf type theory the distinction already clear to Frege between proposition and judgements. Propositions are objects of the theory on which we can make judgements.
  The judgement:

  $$A \textit{ set}$$

  says that A is a set but also that A is a proposition.

- As in natural deduction, derivations are trees where premises and conclusion are judgements.
- We have two kind of judgements.
  - categorical judgements: judgements which do not depend by assumption
  - hypothetical judgements: judgements which are made under assumption

Categorical Judgement

A *set*

To know that A is a set is to know how to form the canonical elements in the set and under what conditions two canonical elements are equal.

$$A = B$$

To know that two sets, A and B, are equal is to know that a canonical element in the set A is also a canonical element in the set B and, moreover, equal canonical elements of the set A are also equal canonical elements of the set B, and vice versa.

$$a \in A$$

If A is a set then to know that $a \in A$ is to know that a, when evaluated, yields a canonical element in A as value.

$$a = b \in A$$

To know that a and b are equal elements in the set A, is to know that they yield equal canonical elements in the set A as values.

Interpretation

Under Curry-Howard correspondence we can read the judgement

$$A \textit{ set}$$

as “A is a proposition”, we can explicitly write this correspondence as

$$A \textit{ prop}$$

Analogously we can consider:

$$a \in A$$

as a proof named a of the proposition A hence read it as “A is true”, we can explicitly write this as

$$A \textit{ true}$$

omitting the proof a.

Hypothetical Judgement

The four categorical judgements can be generalized in order to allow judgements which are made under one or more hypothesis.

$$B(x) \ \textsf{set} \ (x \in A)$$

says that for arbitrary a in the set A, $B(a)$ is a set and if a and b are equal elements in A, then $B(a)$ and $B(b)$ are equal sets.

$B(x)$ is usually said a family of set over A.

The other hypothetical judgements are:

$$B(x) = D(x) \ (x \in A) \qquad b(x) \in B(x) \ (x \in A)$$
$$b(x) = d(x) \in B(x) \ (x \in A)$$

We can also extend notation to an arbitrary number n of assumption.

$$A(x_1, \ldots, x_n) \ \textsf{set} \ (x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$$

Some Derivable Rules - Equality

Assumption:

$$\frac{A \text{ set}}{x \in A \ (x \in A)}$$

Proposition as Set:

$$\frac{x \in A}{A \textit{ true}}$$

Reflexivity:

$$\frac{a \in A}{a = a \in A} \qquad \frac{A \text{ set}}{A = A}$$

Symmetry:

$$\frac{a = b \in A}{b = a \in A} \qquad \frac{A = B}{B = A}$$

Transitivity

$$\frac{a = b \in A \quad b = c \in A}{a = c \in A} \qquad \frac{A = B \quad B = C}{A = C}$$

Equality of set:

$$\frac{a \in A \quad A = B}{a \in B} \qquad \frac{a = b \in A \quad A = B}{a = b \in B}$$

Some Derivable Rules - Substitution

Substitution in sets:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ B(x)\ \textsf{set}\end{array}}{B(a)\ \textsf{set}} \qquad\qquad \frac{a = c \in A \qquad \begin{array}{c}(x \in A)\\ B(x)\ \textsf{set}\end{array}}{B(a) = B(c)}$$

Substitution in equal sets:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ B(x) = D(x)\end{array}}{B(a) = D(a)}$$

Substitution in equal elements:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ b(x) = d(x) \in B(x)\end{array}}{b(a) = d(a) \in B(a)}$$

Substitution in elements:

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ b(x) \in B(x)\end{array}}{b(a) \in B(a)} \qquad\qquad \frac{a = c \in A \qquad \begin{array}{c}(x \in A)\\ b(x) \in B(x)\end{array}}{b(a) = b(c) \in B(a)}$$

Rules

In what follows we will introduce different set forming operation by rules following a common pattern. For each operation we have four kind of rules:

Formation

They say that we can form a certain set from certain other sets or families of sets.

Introduction

They say what are canonical elements and what are equal canonical elements of the set, thus giving the meaning of the operation.

Elimination

They show how we may define functions on the set defined by the introduction rules. They usually are a kind of structural induction rules. They introduce the selector which makes it possible to do pattern-matching and primitive recursion over the elements in the set.

Equality

They describe the equalities which are introduced by computation rules for the selector associated with the set. They relate the introduction and elimination rules by showing how a function defined by means of the elimination rule operates on the canonical elements of the set which are generated by introduction rules.

Equality

In Martin-Löf's type theory there are three different equality relation:

Equality in judgements

this is of two kind:

$A = B$ equality in judgements on sets

$a = b \in A$ equality in judgements on elements

Definitional equality

it is a mere stipulation, a relation between linguistic expression which possible can be read as rewriting rules, we write it as $A \equiv B$

Propositional equality

it reflects at the propositional level the equality in judgements. We will introduce a set $I(A, a, b)$ which is not empty if the judgement $a = b \in A$ is valid.

Cartesian Product of a Family of Sets - 1

We would now introduce the cartesian product of a family of sets:

$$(\Pi x \in A)B(x)$$

Π - formation:

$$\frac{\begin{array}{cc} & (x \in A) \\ A \text{ set} & B(x) \text{ set}\end{array}}{(\Pi x \in A)B(x) \text{ set}} \qquad \frac{\begin{array}{cc} & (x \in A) \\ A = C & B(x) = D(x)\end{array}}{(\Pi x \in A)B(x) = (\Pi x \in C)D(x)}$$

Π - introduction:

$$\frac{\begin{array}{c} (x \in A) \\ b(x) \in B(x)\end{array}}{(\lambda x)b \in (\Pi x \in A)B(x)} \qquad \frac{\begin{array}{c} (x \in A) \\ b(x) = d(x) \in B(x)\end{array}}{(\lambda x)b = (\lambda x)d \in (\Pi x \in A)B(x)}$$

When it is clear from the context we will use higher order notation and simply write $\Pi(A, B)$ and $\lambda(b)$ instead of $(\Pi x \in A)B(x)$ and $(\lambda x)b$.

Cartesian Product of a Family of Sets - 2

We introduce a new constant Ap. The rules are justified by computations
$Ap(c, a)$ is a method to compute a canonical element in $B(a)$:

1. evaluating $c \in \Pi(A, B)$ we obtain a canonical element $(\lambda x)b$ where $b(x) \in B(x)$ $(x \in A)$
2. substituting x with $a \in A$ in b we obtain $b(a) \in B(a)$
3. evaluating $b(a)$ we have a canonical element in $B(a)$.

Π - elimination:

$$\frac{c \in \Pi(A, B) \quad a \in A}{Ap(c, a) \in B(a)} \qquad \frac{c = d \in \Pi(A, B) \quad a = b \in A}{Ap(c, a) = Ap(d, b) \in B(a)}$$

Π - equality:

$$\frac{a \in A \quad \begin{matrix}(x \in A) \\ b(x) \in B(x)\end{matrix}}{Ap((\lambda x)b, a) = b(a) \in B(a)} \qquad \frac{c \in (\Pi x \in A)B(x)}{c = (\lambda x)Ap(c, x) \in \Pi(A, B)}$$

Constant Defined in Terms of the Π Set - 1

If we read the rules for cartesian product of a family of set under the “proposition-as-set” interpretation we have that the new introduced symbol corresponds to universal quantifier:

$$(\forall x \in A)B(x) \equiv (\Pi x \in A)B(x)$$

$\forall$ - formation:

$$\frac{A \text{ set} \qquad \begin{array}{c}(x \in A)\\ B(x) \text{ set}\end{array}}{(\Pi x \in A)B(x) \text{ set}} \qquad \Longrightarrow \qquad \frac{A \text{ set} \qquad \begin{array}{c}(x \in A)\\ B(x) \text{ prop}\end{array}}{(\forall x \in A)B(x) \text{ prop}}$$

$\forall$ - introduction:

$$\frac{\begin{array}{c}(x \in A)\\ b(x) \in B(x)\end{array}}{\lambda(b) \in (\Pi x \in A)B(x)} \qquad \Longrightarrow \qquad \frac{\begin{array}{c}(x \in A)\\ B(x) \text{ true}\end{array}}{(\forall x \in A)B(x) \text{ true}}$$

Constant Defined in Terms of the Π Set - 2

$\forall$ - elimination:

$$\frac{c \in (\Pi x \in A)B(x) \quad a \in A}{Ap(c,a) \in B(a)} \qquad \Longrightarrow \qquad \frac{(\forall x \in A)B(x) \text{ true} \quad a \in A}{B(a) \text{ true}}$$

If in $(\Pi x \in A)B(x)$ the set B do not depends on x, we have the particular case that the introduced set represents the function space from a A to B:

$$A \rightarrow B \equiv (\Pi x \in A)B$$

$\rightarrow$ - formation $\rightarrow$ - introduction $\rightarrow$ - elimination:

$$\frac{A \text{ set} \quad \begin{array}{c}(x \in A)\\ B \text{ set}\end{array}}{A \rightarrow B \text{ set}} \qquad \frac{\begin{array}{c}(x \in A)\\ b(x) \in B\end{array}}{\lambda(b) \in A \rightarrow B} \qquad \frac{f \in A \rightarrow B \quad a \in A}{Ap(f,a) \in B}$$

Constant Defined in Terms of the Π Set - 3

The equality rule corresponds to reduction rules:

$$\frac{a \in A \quad \begin{array}{c}(x \in A)\\ b(x) \in B\end{array}}{Ap(\lambda(b), a) = b(a) \in B}$$

If in $(\Pi x \in A)B(x)$ the set B do not depends on x, under "propositions-as-sets" interpretation we have that the introduced symbol corresponds also to implication:

$$A \supset B \equiv A \rightarrow B \equiv (\Pi x \in A)B$$

$\supset$ - formation

$$\frac{A \text{ prop} \quad \begin{array}{c}(A \text{ true})\\ B \text{ prop}\end{array}}{A \supset B \text{ prop}}$$

$\supset$ - introduction

$$\frac{\begin{array}{c}(A \text{ true})\\ B \text{ true}\end{array}}{A \supset B \text{ true}}$$

$\supset$ - elimination:

$$\frac{A \supset B \text{ true} \quad A \text{ true}}{B \text{ true}}$$

Π Examples

We can define the combinator I:

$$\dfrac{\dfrac{A \text{ set}}{x \in A(x \in A)}}{(\lambda x)x \in A \to A}$$

and also the combinator S:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(x \in A) \quad (f \in A \to B)}{Ap(f,x) \in B} \qquad \dfrac{(x \in A) \quad (g \in A \to (B \to C))}{Ap(g,x) \in B \to C}}{Ap(Ap(g,x), Ap(f,x)) \in C}}{(\lambda x)Ap(Ap(g,x), Ap(f,x)) \in (A \to C)}}{(\lambda f)(\lambda x)Ap(Ap(g,x), Ap(f,x)) \in ((A \to B) \to (A \to C))}}{(\lambda g)(\lambda f)(\lambda x)Ap(Ap(g,x), Ap(f,x)) \in (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

where:

$$\dfrac{A \text{ set}}{x \in A(x \in A)} \qquad \dfrac{A \text{ set} \quad B \text{ set}(x \in A)}{f \in A \to B \ (f \in A \to B)} \qquad \dfrac{A \text{ set} \quad \dfrac{B \text{ set } (x \in A) \quad C \text{ set}(x \in A, y \in B(x))}{B \to C \text{ set}(x \in A)}}{g \in A \to (B \to C) \ (g \in A \to (B \to C))}$$

Disjoint Union of a Family of Sets - 1

We would now introduce the disjoint union of a family of sets:

$$(\Sigma x \in A)B(x)$$

Σ - formation:

$$\frac{A \text{ set} \quad \begin{matrix}(x \in A)\\ B(x) \text{ set}\end{matrix}}{(\Sigma x \in A)B(x) \text{ set}} \qquad \frac{A = C \quad \begin{matrix}(x \in A)\\ B(x) = D(x)\end{matrix}}{(\Sigma x \in A)B(x) = (\Sigma x \in C)D(x)}$$

Σ - introduction:

$$\frac{a \in A \quad b \in B(a)}{\langle a, b\rangle \in (\Sigma x \in A)B(x)} \qquad \frac{a = c \in A \quad b = d \in B(a)}{\langle a, b\rangle = \langle c, d\rangle \in (\Sigma x \in A)B(x)}$$

by higher order notation can we simply write $(\Sigma x \in A)B(x)$ as $\Sigma(A, B)$

Disjoint Union of a Family of Sets - 2

We can now introduce a new constant E. $E(c, d)$ is a method to calculate a canonical element in $B(a)$. The elimination rules justification is in the computation rules represented by equality rules.

Σ - elimination:

$$\frac{c \in (\Sigma x \in A)B(x) \qquad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x, y) \in C(\langle x, y\rangle)\end{array}}{E(c, d) \in C(c)}$$

$$\frac{c = f \in (\Sigma x \in A)B(x) \qquad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x, y) = g(x, y) \in C(\langle x, y\rangle)\end{array}}{E(c, d) = E(f, g) \in C(c)}$$

Σ - equality:

$$\frac{a \in A \qquad b \in B(a) \qquad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x, y) \in C(\langle x, y\rangle)\end{array}}{E(\langle a, b\rangle, d) = d(a, b) \in C(\langle a, b\rangle)}$$

$$\frac{c \in (\Sigma x \in A)B(x)}{c = \langle \pi_0(c), \pi_1(c)\rangle \in (\Sigma x \in A)B(x)}$$

dove $\pi_0(c) \equiv E(c, \lambda(x, y).x)$ e $\pi_1(c) \equiv E(c, \lambda(x, y).y)$.

Constant Defined in Terms of the Σ Set - 1

If we read the rules for disjoint union of a family of set under the "proposition-as-set" interpretation we have that the new introduced symbol corresponds to existential quantifier:

$$(\exists x \in A)B(x) \equiv (\Sigma x \in A)B(x)$$

$\exists$ - formation:

$$\frac{A \text{ set} \qquad \begin{array}{c}(x \in A)\\ B(x) \text{ set}\end{array}}{(\Sigma x \in A)B(x) \text{ set}} \qquad \Longrightarrow \qquad \frac{A \text{ set} \qquad \begin{array}{c}(x \in A)\\ B(x) \text{ prop}\end{array}}{(\exists x \in A)B(x) \text{ prop}}$$

$\exists$ - introduction:

$$\frac{a \in A \quad b \in B(a)}{\langle a, b\rangle \in (\Sigma x \in A)B(x)} \qquad \Longrightarrow \qquad \frac{a \in A \quad B(a) \text{ true}}{(\exists x \in A)B(x) \text{ true}}$$

Constant Defined in Terms of the Σ Set - 2

$\exists$ - elimination:

$$\frac{c \in (\Sigma x \in A)B(x) \qquad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x,y) \in C(\langle x, y\rangle)\end{array}}{E(c,d) \in C(c)} \quad \Longrightarrow \quad \frac{(\exists x \in A)B(x) \textit{ true} \qquad \begin{array}{c}(x \in A, B(x) \textit{ true}) \\ C \textit{ true}\end{array}}{C \textit{ true}}$$

If in $(\Sigma x \in A)B(x)$ the set B do not depends on x we have the particular case that the set represents the cartesian product of A and B:

$$A \times B \equiv (\Pi x \in A)B$$

$\times$ - formation

$$\frac{A \textit{ set} \qquad \begin{array}{c}(x \in A) \\ B \textit{ set}\end{array}}{A \times B \textit{ set}}$$

$\times$ - introduction

$$\frac{a \in A \qquad b \in B}{\langle a, b\rangle \in A \times B}$$

$\times$ - elimination:

$$\frac{c \in A \times B \qquad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x,y) \in C(\langle x, y\rangle)\end{array}}{E(c,d) \in C(c)}$$

Constant Defined in Terms of the Σ Set - 3

The equality rule corresponds to the reduction rule:

$$\frac{a \in A \quad b \in B \quad \begin{array}{c}(x \in A, y \in B(x)) \\ d(x,y) \in C(\langle x,y \rangle)\end{array}}{E(\langle a,b \rangle, d) = d(a,b) \in C(c)}$$

If in $(\Sigma x \in A)B(x)$ the set B do not depends on x then under the "proposition-as-set" interpretation we have that the introduced symbol corresponds to conjunction symbol:

$$A\&B \equiv (\Pi x \in A)B$$

$\&$ - formation

$$\frac{A \text{ prop} \quad \begin{array}{c}(A \text{ true}) \\ B \text{ prop}\end{array}}{A\&B \text{ prop}}$$

$\&$ - introduction

$$\frac{A \text{ true} \quad B \text{ true}}{A\&B \text{ true}}$$

$\&$ - elimination:

$$\frac{A\&B \text{ true} \quad \begin{array}{c}(A \text{ true}, B \text{ true}) \\ C \text{ true}\end{array}}{C \text{ true}}$$

Disjoint Union of two Sets - 1

We define the disjoint union of two sets as primitive

$$A + B$$

because it will be useful to interpret disjunction.

$+$ - formation

$$\frac{A \text{ set} \qquad B \text{ set}}{A + B \text{ set}}$$

$+$ - introduction:

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

$+$ - elimination

$$\frac{c \in A + B \qquad \begin{array}{c}(x \in A)\\ d(x) \in C(inl(x))\end{array} \qquad \begin{array}{c}(y \in B)\\ e(x) \in C(inr(y))\end{array}}{D(c, d, e) \in C(c)}$$

Disjoint Union of two Sets - 2

$+$ - equality

$$\frac{a \in A \quad \begin{array}{c}(x \in A)\\ d(x) \in C(inl(x))\end{array} \quad \begin{array}{c}(y \in B)\\ e(x) \in C(inr(y))\end{array}}{D(inl(a), d, e) = d(a) \in C(inl(a))}$$

$$\frac{b \in B \quad \begin{array}{c}(x \in A)\\ d(x) \in C(inl(x))\end{array} \quad \begin{array}{c}(y \in B)\\ e(x) \in C(inr(y))\end{array}}{D(inr(b), d, e) = d(a) \in C(inr(b))}$$

We can now set : $A \vee B \equiv A + B$

$\vee$ - formation

$$\frac{A \text{ prop} \quad B \text{ prop}}{A \vee B \text{ prop}}$$

$\vee$ - introduction

$$\frac{A \text{ true}}{A \vee B \text{ true}} \qquad \frac{B \text{ true}}{A \vee B \text{ true}}$$

$\vee$ - elimination:

$$\frac{A \vee B \text{ true} \quad \begin{array}{c}(A \text{ true})\\ C \text{ true}\end{array} \quad \begin{array}{c}(B \text{ true})\\ C \text{ true}\end{array}}{C \text{ true}}$$

(Extensional) Propositional Equality

We would now introduce an equality relation $I(A, a, b)$ in order to reflect equality in judgements at propositional level.

I - formation

$$\frac{A \textsf{ set} \quad a \in A \quad b \in A}{I(A, a, b) \textsf{ set}}$$

I - introduction:

$$\frac{a = b \in A}{r \in I(A, a, b)} \qquad \frac{a = b \in A}{r = r \in I(A, a, b)}$$

We could introduce an elimination rules which corresponds to a structural induction principle, instead we prefer follow Martin-Löf and introduce the following as primitive.

I - elimination

$$\frac{c \in I(A, a, b)}{a = b \in A}$$

I - equality

$$\frac{c \in I(A, a, b)}{c = r \in I(A, a, b)}$$

It is interesting to note that I - introduction is the first rule useful to introduce family of sets.

With the introduction of extensional equality we can no longer look at equality in judgement as convertibility, hence it becomes not decidable.

Undecidability of equality in judgements

We show it informally, consider the representation $\hat{f}$ of the function:

$$f(e,t) = \begin{cases} 1, & \text{if TM } e \text{ applied to itself halts after less than t steps with result } 0 \\ 0, & \text{otherwise} \end{cases}$$

If e applied to itself halts after some number of steps with a result other than 0, then $f(e,-)$ is constant 0 and since this fact is provable (by induction and case distinction) in type theory then the set $I(N, \hat{f}\hat{e}t, 0)\ (t \in N)$ is inhabited hence by I - elimination

$$\hat{f}\hat{e}t = 0 \in N\ (t \in N)$$

holds, otherwise if e applied to itself halts with result 0 it do not holds. Suppose equality in judgements is decidable we have a TM e_0 that applied to some number e return 0 if $\hat{f}\hat{e}t = 0 \in N\ (t \in N)$ and 1 otherwise. Considering the application of e_0 to itself we have a contraddiction.

$$\hat{e_0}\hat{e_0} = 0 \iff \hat{f}\hat{e_0}t = 0 \in N\ (t \in N) \iff \not\vdash \hat{f}\hat{e}t = 0 \in N\ (t \in N)$$
$$\hat{e_0}\hat{e_0} = 1 \iff \not\vdash \hat{f}\hat{e_0}t = 0 \in N\ (t \in N) \iff \hat{f}\hat{e}t = 0 \in N\ (t \in N)$$

Examples of Propositional Equality

We can justify the introductory axiom of identity.

$$\dfrac{\dfrac{\dfrac{\dfrac{A \text{ set}}{x \in A \; (x \in A)}}{x = x \in A}}{r \in I(A,x,x)}}{(\lambda x) r \in (\forall x \in A) I(A,x,x)}$$

and also the eliminatory axiom of identity (Leibniz's Principle):

$$\dfrac{\dfrac{\dfrac{\dfrac{(w \in B(x)) \qquad \dfrac{\dfrac{(z \in I(A,x,y))}{x = y \in A} \qquad B(x) \text{ set } (x \in A)}{B(x) = B(y)}}{w \in B(y)}}{(\lambda w) w \in B(x) \supset B(y)}}{(\lambda z)(\lambda w) w \in I(A,x,y) \supset (B(x) \supset B(y))}}{(\lambda x)(\lambda y)(\lambda z)(\lambda w) w \in (\forall x \in A)(\forall y \in A)(I(A,x,y) \supset (B(x) \supset B(y)))}$$

Finally (finite) Set

We can now introduce the finite sets, the existence of these objects is a primitive assumption, hence the formation rules will no have assumption, furthermore we will give an infinite number of rules, one for each $n \in N$.

N_n - formation

$$N_n \textit{ set}$$

N_n - introduction:

$$m_n \in N_n \quad (m = 0, 1, \ldots, n-1)$$

We clearly have that N_0 has no elements, N_1 has a single canonical element 0_1, N_2 has two canonical elements $0_2, 1_2$, etc.

N_n - elimination

$$\frac{c \in N_n \quad c_m \in C(m_n) \quad (m = 0, 1, \ldots, n-1)}{R_n(c, c_0, \ldots, c_{n-1}) \in C(c)}$$

N_n - equality:

$$\frac{c_m \in C(m_n) \quad (m = 0, 1, \ldots, n-1)}{R_n(m_n, c_0, \ldots, c_{n-1}) = c_m \in C(c)}$$

R_n represents a kind of definition by cases.

Examples of Finite Sets

N_0 do not have introduction rules hence it do not have elements, we can set:

$$\perp \equiv \emptyset \equiv N_0$$

and it follows that the elimination rule represents “ex falso quodlibet”

$$\frac{c \in N_0}{R_0(c) \in C(c)} \qquad \Longrightarrow \qquad \frac{\perp \ \textit{true}}{C \ \textit{true}}$$

we can hence use N_0 to introduce negation

$$\neg A \equiv A \rightarrow N_0$$

In the same way we can interpret:

$$Bool \equiv N_2$$

$$\mathsf{tt} \equiv 0_2 \qquad \mathsf{ff} \equiv 1_2 \qquad \text{if } c \text{ then } c_1 \text{ else } c_2 \equiv R(c, c_1, c_2)$$

Natural Number - 1

We now introduce the first infinite set.

N - formation

$$N \textit{ set}$$

N - introduction:

$$0 \in N \qquad \frac{n \in N}{s(n) \in N}$$

N - elimination

$$\frac{c \in N \quad d \in C(0) \quad \begin{array}{c}(x \in N, y \in C(x)) \\ e(x,y) \in C(s(x))\end{array}}{R(c,d,e) \in C(c)}$$

N - equality:

$$\frac{d \in C(0) \quad \begin{array}{c}(x \in N, y \in C(x)) \\ e(x,y) \in C(s(x))\end{array}}{R(0,d,e) = d \in C(0)} \qquad \frac{n \in N \quad d \in C(0) \quad \begin{array}{c}(x \in N, y \in C(x)) \\ e(x,y) \in C(s(x))\end{array}}{R(s(n),d,e) = e(n, R(n,d,e)) \in C(s(n))}$$

Natural Number - 2

If we interpret $C(z)(z \in N)$ as a propositional function we have:

N - elimination

Mathematical Induction

$$\frac{c \in N \quad d \in C(0) \quad \begin{array}{c}(x \in N, y \in C(x))\\ e(x,y) \in C(s(x))\end{array}}{R(c,d,e) \in C(c)} \quad \Longrightarrow \quad \frac{c \in N \quad C(0)\ \textit{true} \quad \begin{array}{c}(x \in N, C(x)\ \textit{true})\\ C(s(x))\ \textit{true}\end{array}}{C(c)\ \textit{true}}$$

As usual with R we can define the usual function over natural number, ex:

$$\mathsf{pd}(n) \equiv R(n, 0, \lambda(x,y).x) \qquad n + m \equiv R(n, 0, \lambda(x,y).s(y))$$
$$n \times m \equiv R(n, s(o), \lambda(x,y).m + y)$$

and derive the usual rules, ex:

$$\frac{a \in N \quad b \in N}{a \times b \in N} \qquad \frac{s(a) = s(b) \in N}{a = b \in N} \qquad \frac{a \in N \quad b \in N}{a + s(b) = s(a + b) \in N}$$

Lists

List - formation

$$\frac{A \text{ set}}{List(A) \text{ set}}$$

List - introduction

$$nil \in List(A)$$

$$\frac{a \in A \quad b \in List(A)}{(a.b) \in List(A)}$$

List - elimination:

$$\frac{c \in List(A) \quad d \in C(nil) \quad \begin{array}{c}(x \in A, y \in List(A), z \in C(y)) \\ e(x,y,z) \in C((x.y))\end{array}}{listrec(c,d,e) \in C(c)}$$

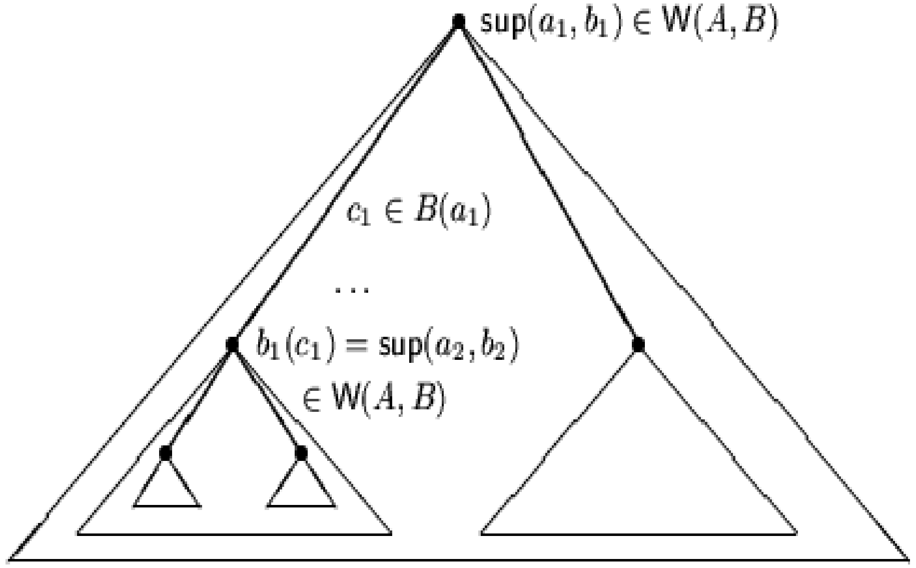
List - equality:

$$\frac{a \in A \quad b \in List(A) \quad d \in C(nil) \quad \begin{array}{c}(x \in A, y \in List(A), z \in C(y)) \\ e(x,y,z) \in C((x.y))\end{array}}{listrec((a.b),d,e) = e(a,b,listrec(b,d,e)) \in C((a.b))}$$

$$\frac{d \in C(nil) \quad \begin{array}{c}(x \in A, y \in List(A), z \in C(y)) \\ e(x,y,z) \in C((x.y))\end{array}}{listrec(nil,d,e) = d \in C(nil)}$$

Wellorderings - 1

We would now introduce a constructor in order to define wellorderings set. We can consider a wellordering as a well founded tree:



$\mathsf{sup}(a_1, b_1) \in \mathsf{W}(A, B)$
$c_1 \in B(a_1)$
$\dots$
$b_1(c_1) = \mathsf{sup}(a_2, b_2)$
$\in \mathsf{W}(A, B)$

We need to know:

- the different ways the tree may be formed
- for each way to form a tree which parts it consists of

Wellorderings - 2

We would introduce a constructor

$$(Wx \in A)B(x)$$

which we will simply write as $W(A, B)$.

W - formation

$$\frac{A \text{ set} \qquad \begin{array}{c}(x \in A)\\ B(x) \text{ set}\end{array}}{W(A, B) \text{ set}}$$

W -introduction

$$\frac{a \in A \qquad b \in B(a) \to W(A, B)}{sup(a, b) \in W(A, B)}$$

$$\frac{c \in W(A, B) \qquad \begin{array}{c}(x \in A, y \in B(x) \to W(A, B), z \in (\Pi v \in B(x))C(app(y, v)))\\ b(x, y, z) \in C(sup(x, y))\end{array}}{wrec(c, b) \in C(c)}$$

Wellorderings - 3

W - equality

$$\frac{a \in A \quad b \in B(a) \to W(A,B) \quad \begin{array}{c}(x \in A, y \in B(x) \to W(A,B), z \in (\Pi v \in B(x))C(app(y,v)))\\ d(x,y,z) \in C(sup(x,y))\end{array}}{wrec(sup(a,b),d) = d(a,b,(x)wrec(b(x),d)) \in C(sup(a,b))}$$

Under the “proposition-as-set” interpretation we have that W - elimination rule represents transfinite induction principle:

$$\frac{c \in W(A,B) \quad \begin{array}{c}(x \in A, y \in B(x) \to W(A,B), z \in (\Pi v \in B(x))C(app(y,v)))\\ b(x,y,z) \in C(sup(x,y))\end{array}}{wrec(c,b) \in C(c)}$$

$$\frac{c \in W(A,B) \quad \begin{array}{c}(\forall x \in A)(\forall y \in B(x) \to W(A,B))\\ ((\forall v \in B(x))C(app(y,v)) \supset C(sup(x,y)))\ \textit{true}\end{array}}{C(c)\ \textit{true}}$$

Example of Wellorderings

Instead of taking the set of Natural Numbers as primitive we can define it as a wellorderings:

$$N \equiv (Wx \in N_2)B(x)$$

where $B(x)$ $(x \in N_2)$ is a family of sets such that:

$$B(0_2) = N_0 \qquad\qquad B(1_2) = N_1$$

hence we can for example define:

$$zero \equiv sup(0_2, \lambda x.\bot)$$

and

$$succ(n) \equiv sup(1_2, \lambda x.n)$$

Wellorderings are useful for define inductive data type, in particular they can be used also to define ordinal classes.

Set of Small Set - First Universe - 1

- We have introduced set forming operation which can be finitely iterated.
- We need a way to reflect set structure at object level in order to be able to infinitely iterate set forming operations.
- The idea is to define a universe as the least set closed under certain specified set forming operations.

We now introduce the set $\mathbf{U}$ of small sets (first universe) as the least set closed under the set forming operation we have introduced.

$\mathbf{U}$ - formation

$$\mathbf{U} \ \textit{set}$$

and a family of set useful to decode.

$\mathbf{Set}$ - formation

$$\frac{A \in \mathbf{U}}{\mathbf{Set}(A) \ \textit{set}} \qquad \frac{A = B \in \mathbf{U}}{\mathbf{Set}(A) = \mathbf{Set}(B)}$$

Set of Small Set - First Universe - 2

U - introduction

$$\frac{A \in \mathbf{U} \qquad \overset{(x \in \mathbf{Set}(A))}{B(x) \in \mathbf{U}}}{\widehat{\Pi}(A, B) \in \mathbf{U}}$$

$$\frac{A \in \mathbf{U} \qquad \overset{(x \in \mathbf{Set}(A))}{B(x) \in \mathbf{U}}}{\widehat{\Sigma}(A, B) \in \mathbf{U}}$$

$$\frac{A \in \mathbf{U} \qquad B \in \mathbf{U}}{A \widehat{+} B \in \mathbf{U}}$$

$$\frac{A \in \mathbf{U} \qquad a \in \mathbf{Set}(A) \qquad b \in \mathbf{Set}(A)}{\widehat{I}(A, a, b) \in \mathbf{U}}$$

Set - introduction

$$\frac{A \in \mathbf{U} \qquad \overset{(x \in \mathbf{Set}(A))}{B(x) \in \mathbf{U}}}{\mathbf{Set}(\widehat{\Pi}(A, B)) = \Pi(\mathbf{Set}(A), \lambda(x)\mathbf{Set}(B(x)))}$$

$$\frac{A \in \mathbf{U} \qquad \overset{(x \in \mathbf{Set}(A))}{B(x) \in \mathbf{U}}}{\mathbf{Set}(\widehat{\Sigma}(A, B)) = \Sigma(\mathbf{Set}(A), \lambda(x)\mathbf{Set}(B(x)))}$$

$$\frac{A \in \mathbf{U} \qquad B(x) \in \mathbf{U}}{\mathbf{Set}(A \widehat{+} B) = \mathbf{Set}(A) + \mathbf{Set}(B)))}$$

$$\frac{A \in \mathbf{U} \qquad a \in \mathbf{Set}(A) \qquad b \in \mathbf{Set}(B)}{\mathbf{Set}(\widehat{I}(A, a, b)) = I(\mathbf{Set}(A), a, b)}$$

Set of Small Set - First Universe - 3

U - introduction

$$\widehat{N_0} \in \mathbf{U} \qquad \widehat{N_1} \in \mathbf{U} \cdots$$

$$\widehat{N} \in \mathbf{U}$$

$$\frac{A \in U}{\widehat{List}(A) \in \mathbf{U}}$$

$$\frac{A \in \mathbf{U} \qquad \begin{array}{c}(x \in \mathbf{Set}(A)) \\ B(x) \in \mathbf{U}\end{array}}{\widehat{W}(A, B) \in \mathbf{U}}$$

Set - introduction

$$\mathbf{Set}(\widehat{N_0}) = N_0 \qquad \mathbf{Set}(\widehat{N_1}) = N_1 \ \cdots$$

$$\mathbf{Set}(\widehat{N}) = N$$

$$\frac{A \in \mathbf{U}}{\mathbf{Set}(\widehat{List}(A)) = List(\mathbf{Set}(A))}$$

$$\frac{A \in \mathbf{U} \qquad \begin{array}{c}(x \in \mathbf{Set}(A)) \\ B(x) \in \mathbf{U}\end{array}}{\mathbf{Set}(\widehat{W}(A, B)) = W(\mathbf{Set}(A), \lambda(x)\mathbf{Set}(B(x)))}$$

We could iterate the process, obtaining a second universe $\mathbf{U}'$, a third $\mathbf{U}''$, and so on.

$$\widehat{U} \in \mathbf{U}' \qquad \mathbf{Set}'(\widehat{U}) = \mathbf{U} \qquad \frac{A \in \mathbf{U}}{\widehat{\mathbf{Set}}(A) \in \mathbf{U}'} \qquad \frac{A \in \mathbf{U}}{\mathbf{Set}'(\widehat{\mathbf{Set}}(A)) = \mathbf{Set}(A)}$$

Example of Universe

We would prove the fourth Peano axiom, not derivable in type theory without universes.

$$(\forall n \in N)0 \neq s(n)$$

Remember that $(\forall n \in N)0 \neq s(n) \equiv (\forall n \in N)I(N, 0, s(n)) \rightarrow N_0$. If we set $NZ(x) \equiv R(x, \widehat{N_0}, (m, n)\widehat{N_1})$ and we leave implicit some assumption we have:

$$\dfrac{\dfrac{0_1 \in N_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}(x \in N)\\(y \in I(N, 0, s(x)))\end{array}}{0 = s(x) \in N} \qquad \widehat{N_0} \in \mathbf{U} \quad \widehat{N_1} \in \mathbf{U}}{NZ(0) = NZ(s(x)) \in \mathbf{U}} \qquad NZ(0) = \widehat{N_0} \quad NZ(s(x)) = \widehat{N_1}}{\widehat{N_0} = \widehat{N_1} \in \mathbf{U}}}{\mathbf{Set}(\widehat{N_0}) = \mathbf{Set}(\widehat{N_1})}}{N_0 = N_1}}{0_1 \in N_0}}{(\lambda y)0_1 \in I(N, 0, s(x)) \rightarrow N_0}}{(\lambda x)(\lambda x)0_1 \in (\forall x \in N)I(N, 0, s(x)) \rightarrow N_0}$$

Types

- We have introduced a collection of sets, set forming operations and introduced proof rules for these sets.
- Another way to introduce sets is to use the primitive notion of types.

Remember the definition of set:

> What does it mean that A is a set? To know that A is a set is to know how to form canonical object of A, as well as what it means for two canonical objects to be the same.

Now we have:

> What does it mean that A is a type? To know that A is a type is to know what it means to be an object of A, as well as what it means for two objects to be the same.

The judgements now are:

$$A \text{ Type} \qquad\qquad A = B$$

$$a : A \qquad\qquad a = b : A$$

Type of Set - 1

By the explanation of set:

> What does it mean that A is a set? To know that A is a set is to know how to form canonical object of A, as well as what it means for two canonical objects to be the same.

we can also derive:

Set -formation:

$$\textit{Set}\ \textit{Type}$$

and also:

El - formation:

$$\frac{A : \textit{Set}}{El(A)\ \textit{Type}} \qquad \frac{A = B : \textit{Set}}{El(A) = El(B)}$$

where $El(A)$ is the type of the elements of the set A.

Type of Set - 2

Note that *Set* is an open concept, we have not exhausted the possibilities of defining new sets. In contrast a set is always an inductive (closed) structure. For example $\mathbf{U}$ is a closed concept, whose canonical elements are coding of a fixed number of set constructing operations.

In presentation of types for clarity we have changed the notation, nevertheless we can recover the previous notation by the following definitions:

$$A \textsf{ set} \equiv A \in \textsf{Set} \equiv A : \textsf{Set} \qquad\qquad a \in A \equiv a : El(A)$$

And when it is clear we will write A instead of $El(A)$.

We can adapt the definition given for set and define:

- families of types (we write $A \ \textsf{Type}\ [x : A_1]$)
- rules for equality
- rules for substitution

Assumption Rule

The assumption rule is more interesting. Remember that for set we had the assumption rule:

$$\frac{C \text{ set}}{x \in C \; (x \in C)}$$

for type we clearly have:

Assumption

$$\frac{C \text{ Type}}{x : C \; [x : C]}$$

Note that the above is more general, in fact we can now have two kind of conclusion:

$$\frac{\textit{Set Type}}{X : \textit{Set} \; [X : \textit{Set}]} \quad \Longrightarrow \quad \frac{\textit{Set Type}}{X \text{ set} \; [X \text{ set}]}$$

$$\frac{\dfrac{\dfrac{\textit{Set Type}}{X : \textit{Set} \; [X : \textit{Set}]}}{El(X) \textit{ Type} \; [X : \textit{Set}]}}{x : El(X) \; [X : \textit{Set}, x : El(X)]} \quad \Longrightarrow \quad \frac{\dfrac{\dfrac{\textit{Set Type}}{X \text{ set} \; [X \text{ set}]}}{X \textit{ Type} \; [X \text{ set}]}}{x \in X \; [X \text{ set}, x \in X]}$$

Function Types - 1

We now would introduce a way to define compound type. We can define function type.

$$(x \in A)B$$

To know that an object c is of type $(x \in A)B$ means that we know that when we apply it to an arbitrary object a of type A we get an object $c(a)$ in $B[x \leftarrow a]$, [. . .].

Function type (formation):

$$\frac{A \text{ Type} \qquad \begin{array}{c}[x \in A]\\ B \text{ Type}\end{array}}{(x \in A)B \text{ Type}} \qquad\qquad \frac{A = A' \qquad \begin{array}{c}[x \in A]\\ B = B'\end{array}}{(x \in A)B = (x \in A')B'}$$

Application (elimination):

$$\frac{c \in (x \in A)B \qquad a \in A}{c(a) \in B[x \leftarrow a]} \qquad\qquad \frac{c \in (x \in A)B \qquad a = b \in A}{c(a) = c(b) \in B[x \leftarrow a]}$$

Function Types - 2

Functions can be formed by abstraction and the rule for abstraction is justified by conversion:

Abstraction (introduction):

$$\frac{\begin{matrix}[x \in A] \\ b \in B\end{matrix}}{[x]b \in (x \in A)B}$$

β-conversion (equality):

$$\frac{a \in A \qquad \begin{matrix}[x \in A] \\ b \in B\end{matrix}}{([x]b)(a) = b[x \leftarrow a] \in B[x \leftarrow a]}$$

Note that application is more primitive then abstraction on type level.

By the rules we have introduced we can derive :

η-conversion:

$$\frac{c \in (x \in A)B}{([x]c)(x) = c \in (x \in A)B}$$

ξ-rule :

$$\frac{\begin{matrix}[x \in A] \\ b = d \in B\end{matrix}}{[x]b = [x]d \in (x \in A)B}$$

Hypothetical Judgements and Functions

From the judgement as $a \in A\ [x_1 \in A_1, x_2 \in A_2, \ldots, x_n \in A_n]$ we can derive, by repeated abstractions judgements as $[x_1, x_2, \ldots, x_n]a \in (x_1 \in A_1)(x_2 \in A_2) \cdots (x_n \in A_n)A$.
Conversely we obtain the former by the latter by repeated assumptions and applications.

$$\cfrac{\cfrac{a \in A\ [x_1 \in A_1, x_2 \in A_2]}{[x_2]a \in (x_2 \in A_2)A\ [x_1 \in A_1]}}{[x_1, x_2]a \in (x_1 \in A_1)(x_2 \in A_2)A} \qquad \cfrac{\cfrac{[x_1]a \in (x_1 \in A_1)A \quad \cfrac{A\ \textit{Type}}{x_1 \in A_1\ [x_1 \in A]}}{([x_1]a)(x_1) \in A[x_1 \leftarrow x_1]\ [x_1 \in A]}}{a \in A\ [x_1 \in A]}$$

We use some notational convention: $f(a, b)$ instead of $f(a)(b)$, $[x, y]e$ instead of $[x][y]e$, $(A)B$ instead of $(x \in A)B$ when B does not depends on x.

By the above we can derive more general assumption as follows:

$$Y(x)\ \textit{set}[X\ \textit{set}, Y(x)\ \textit{set}[x \in X], x \in X]$$

which can be read as:

Assume that $Y(x)$ is a set under the assumptions that X is a set and $x \in X$.

Defining Sets in term of Types - Π

The Π-sets can be introduced by defining the following constants.

$$\begin{aligned}
\Pi &\in (X \in \textsf{Set}, (El(X))\textsf{Set})\textsf{Set} \\
\lambda &\in (X \in \textsf{Set}, Y \in (El(X))\textsf{Set}, (x \in El(X))El(Y(x)))El(\Pi(X,Y)) \\
Ap &\in (X \in \textsf{Set}, Y \in (El(X))\textsf{Set}, El(\Pi(X,Y)), x \in El(X))El(Y(x))
\end{aligned}$$

and asserting the equality $Ap(X,Y,\lambda(X,Y,b),a) = b(a) \in El(Y(a))$. We can write them also as derived rules:

$$\frac{X\textsf{Set} \quad Y(x)\textsf{Set}\ [x \in El(X)]}{\Pi(X,Y)\textsf{Set}} \qquad \frac{X\textsf{Set} \quad Y(x)\textsf{Set}\ [x \in El(X)] \quad b(x) \in El(Y(x))\ [x \in El(X)]}{\lambda(X,Y,b) \in El(\Pi(X,Y))}$$

$$\frac{X\textsf{Set} \quad Y(x)\textsf{Set}\ [x \in El(X)] \quad b \in El(\Pi(X,Y)) \quad x \in El(X)}{Ap(X,Y,b,x) \in El(Y(x))}$$

$$\frac{X\textsf{Set} \quad Y(x)\textsf{Set}\ [x \in El(X)] \quad b \in El(Y(x))\ [x \in El(x)] \quad a \in El(x)}{Ap(X,Y,\lambda(X,Y,b),a) = b(a) \in El(Y(a))}$$

Defining Sets in term of Types - Σ

The Σ-sets can be introduced by defining the following constants.

$$\begin{aligned}
\Sigma &\in (X \in \mathsf{Set}, (El(X))\mathsf{Set})\mathsf{Set} \\
pair &\in (X \in \mathsf{Set}, Y \in (El(X))\mathsf{Set}, x \in El(X), (x \in El(X))El(Y(x)))El(\Sigma(X,Y)) \\
split &\in (X \in \mathsf{Set}, Y \in (El(X))\mathsf{Set}, Z \in (El(\Sigma(X,Y)))\mathsf{Set}, \\
&\quad z \in (x \in El(X), y \in El(Y(x)))El(Z(pair(X,Y,x,y))), \\
&\quad w \in El(\Sigma(X,Y))) \\
&\quad El(Z(w))
\end{aligned}$$

and asserting the equality

$$split(X,Y,Z,z,pair(X,Y,x,y)) = z(x,y) \in El(Z(pair(X,Y,x,y)))$$

where:

$$\begin{array}{cccc}
X \in \mathsf{Set} & Y \in (El(X))\mathsf{Set} & Z \in (El(\Sigma(X,Y)))\mathsf{Set} & x \in El(X) \\
y \in El(Y(x)) & \multicolumn{3}{c}{z \in (x \in El(Z), y \in El(Y(x)))El(Z(pair(X,Y,x,y)))}
\end{array}$$

Defining Sets in term of Types - N

The set of natural number can be introduced by defining the following constants.

$$\begin{array}{rcl}
N & \in & \textsf{Set} \\
0 & \in & El(N) \\
succ & \in & (El(N))El(N) \\
natrec & \in & (Z \in (El(N))\textsf{Set}, \\
 & & \quad z \in El(Z(0)), \\
 & & \quad s \in (x \in El(N), El(Z(x)))El(Z(succ(x))), \\
 & & \quad n \in El(N)) \\
 & & \quad El(Z(n))
\end{array}$$

and asserting the equalities

$$\begin{array}{rcl}
natrec(Z, z, s, 0) & = & z \in El(Z(0)) \\
natrec(Z, z, s, succ(n)) & = & s(n, natrec(Z, z, s, n) \in El(Z(succ(n)))
\end{array}$$

Extensionality Vs Intensionality

We have so far introduced extensional equality I. In the type theory we are developing we cannot derive elimination rule for I, hence it do not fit in this type theory. Instead of the extensional equality I we can introduce an intensional equality Id by defining the following constant:

$$\begin{aligned}
Id &\in (X \in \textsf{Set}, El(X), El(X))\textsf{Set} \\
id &\in (X \in \textsf{Set}, x \in El(X))Id(X, x, x) \\
idpeel &\in (X \in \textsf{Set}, x \in El(X), y \in El(X), \\
&\quad Z \in (m \in El(X), n \in El(X), El(Id(X, m, n)))\textsf{Set}, \\
&\quad v \in (z \in El(X))El(Z(z, z, id(X, z))), \\
&\quad u \in El(Id(X, x, y)), \\
&\quad El(Z(x, y, u))
\end{aligned}$$

and asserting the equality:

$$idpeel(X, x, y, Z, v, id(X, x)) = v(x) \in El(Z(x, x, id(X, x)))$$

Polymorphic Vs Monomorphic

The first type theory we have so far introduced is a polymorphic type theory while the second is monomorphic. Consider the constant Ap, in the polymorphic version we have:

$$Ap(f, a)$$

while in the monomorphic we have:

$$Ap(A, B, f, a)$$

Hence in the monomorphic type theory we have that all important information about the validity of a judgement is contained in the judgement itself. Hence, given a judgement, it is possible to reconstruct a derivation of the judgement (type checking is decidable).
We can define a stripping function which erase type information in the monomorphic theory:

> Nevertheless, the polymorphic theory is fundamentally different from the monomorphic theory in the sense that there are derivable judgements in the polymorphic theory which cannot come from any derivable judgement in the monomorphic theory by stripping.

Bibliografia

- Per Martin-Löf,Intuitionistic Type Theory. Bibliopolis, 1984.
- Bengt Nordström, Kent Petersson and Jan M. Smith, Programming in Martin-Löf's Type Theory: an introduction. Oxford University Press, 1990. http://www.cs.chalmers.se/Cs/Research/Logic/book/
- Bengt Nordström, Kent Petersson and Jan M. Smith, Martin-Löf's Type Theory, Chapter in Handbook of Logic in Computer Science, Vol 5, Oxford University Press, 2000. http://www.cs.chalmers.se/ bengt/papers/hlcs.ps
- Laura Crosilla, Introduzione alla Teoria dei Tipi di Martin-Löf. Appunti per le lezioni tenute in occasione del seminario "Tipi e Informazione"; Dipartimento di Filosofia, Università di Firenze, 2003.