Models of Type Theory

Thierry Coquand

August. 22, 2007

Content of the course

Lecture I: Short history of type theory, connections set theory/type theory

Lecture II: Models of type theory, category with families, applications of semantics

Content of the course

Lecture II: presentation of the D-set model (Hyland, Moggi)

M. Hofmann *Syntax and Semantics of Dependent Types*

G. Longo and E. Moggi *Constructive Natural Deduction and its "ω-set" interpretation*

Applications: (relative) consistency proof for a system without set-theoretic models

non derivability results (Σ is not definable in CC)

Lecture I: slides of Alexandre Miquel, TYPES Summer School 2005

Foundations of mathematics

type theory (1908), simple type theory (1940), constructive type theory (1972)

set theory (1908), Z, ZF, ZFC, BG

topos theory (1970)

Several alternatives: extensional/intensional, predicative/impredicative, classical/intuitionistic, axiom of choice

An optimal type system?

Martin-Löf (1970): $\lambda*$

$$M, A \ ::= \ x \mid \lambda x : A.M \mid \Pi x : A.A \mid *$$

$$\frac{}{() \text{ wf}} \qquad \frac{\Gamma \vdash A : *}{\Gamma, x : A \text{ wf}}$$

An optimal type system?

$$\frac{\Gamma \text{ wf} \quad x : A \; in \; \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash N : \Pi x : A.B \quad \Gamma \vdash M : A}{\Gamma \vdash N \; M : B[M]} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

$$\frac{\Gamma, x : A \vdash B : *}{\Gamma \vdash \Pi x : A.B : *} \qquad \frac{\Gamma \text{ wf}}{\Gamma \vdash * : *}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : * \quad A =_\beta B}{\Gamma \vdash M : B}$$

An optimal type system?

One can interpret $\lambda 2$ in this system

One can interpret higher-order logic, Leibniz equality

One can represent the type of Church numeral $N : *$ and prove the axiom of infinity

An optimal type system?

Girard (1970): this system is contradictory

Not so obvious: *types are not sets*

I present later a possible explanation (A. Miquel, 2001)

T. Hurkens (1994) found a clever short derivation

One can define $Y_k : \Pi A : *.(A \to A) \to A$ such that $Y_k\ A\ f = f\ (Y_{k+1}\ A\ f)$ and hence represent all partial recursive functions $N \to N$ (M. Reinholt, A. Meyer 1986)

Curry-Howard correspondance

Common foundation to *logic* and *set theory*

propositions are identified to *sets*

truth $=$ inhabitation

$\exists i : I.B_i$ is explained as $\Sigma_{i \in I} B_i$

Normalisation proof for $F\omega$

$C_{A\to B}(N) =_{def} \forall M.C_A(M) \Rightarrow C_B(N\ M)$

for $A, B : *$ and $C_A(M)$ is a *proposition*

$C_{*\to *}(N) =_{def} \Pi M.C_*(M) \to C_*(N\ M)$

$C_*(A) =$ *set* of reducibility candidates at type A

$C_\kappa(M)$ is a *set* if κ is a kind

Similar constructions, replacing propositions by sets

Impredicativity and Curry-Howard

Impredicativity: propositions form a type

Curry-Howard: propositions $=$ types

The natural conclusion is that there is a type of all types $* : *$

Impredicativity and Curry-Howard

What goes wrong?

axiom of reducibility: purely pragmatic justification for Russell-Whitehead (1908)

Weyl (1946): this "is a bold, an almost fantastic axiom; there is little justification for it in the real world in which we live, and none at all in the evidence on which our mind bases its constructions"

Predicative type theory

Keep "propositions=types" but take away impredicativity

Keep the idea of *universes*

Replace the universe $*$ by an universe U of *small types*

This was introduced by analogy to the notion of universe previously introduced by Grothendieck

Predicative type theory

We have types Nat, N_0, N_1, N_2

Since the system is predicative we have to introduce these types as *primitive*

Type formations: $\Pi x : A.B$, $\Sigma x : A.B$, $W x : A.B$

Functional programming language with dependent types

How to build dependent types? For instance to define $T\ 0 = \mathsf{N}_0$, $T\ 1 = \mathsf{N}_1$

We need *universes*

Predicative type theory

We replace the type of all type by a universe U of small types

This is a *reflection* principle. We reflect the general construction on types in the type U

$$\frac{A : \mathsf{U}, \quad B : \mathsf{U}\ [x : A]}{\Pi x : A.B\ : \mathsf{U}}, \qquad \frac{A : \mathsf{U}}{A\ \mathsf{type}}$$

Predicative type theory

```
data Nat : Set where
  zero : Nat
  succ  : Nat -> Nat

data _*_ (A B : Set) : Set where
  _,_ : A -> B -> A * B

data N0 : Set where

exit : {A : Set} -> N0 -> A
exit ()
```

Predicative type theory

```
data _+_ (A B : Set) : Set where
  Inl : A -> A + B
  Inr : B -> A + B

_<->_ : Set -> Set -> Set
A <-> B = (A -> B) * (B -> A)

neg : Set -> Set
neg A = A -> NO
```

Predicative type theory

```
data N2 : Set where
  zero : N2
  one  : N2

Vec : Nat -> Set -> Set
Vec zero X = One
Vec (suc n) X = X * Vec n X

data Sigma (A : Set) (B : A -> Set) : Set where
  exI : (a : A) -> B a -> Sigma A B
```

Predicative type theory

```
prop : (F : N2 -> Set) -> F zero + F one <-> Sigma N2 F
prop F = dir , conv
  where
   dir : F zero + F one -> Sigma N2 F
   dir (Inl a0) = exI zero a0
   dir (Inr a1) = exI one  a1

   conv : Sigma N2 F -> F zero + F one
   conv (exI zero a0) = Inl a0
   conv (exI one  a1) = Inr a1
```

Minimal simple type theory

First designed by Church 1940 to simplify the system of Russell-Whitehead, using classical logic

Minimal version: probably first explicitated by Martin-Löf and Girard (1970)

Types $A \ ::= \ A \to A \mid \mathsf{Prop}$

Prop is the type of *propositions*

Terms: simply typed terms with two constants

$(\Rightarrow) : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}, \quad \forall : (A \to \mathsf{Prop}) \to \mathsf{Prop}$

Minimal simple type theory

We write $\forall x : A.\phi$ for $\forall\ (\lambda x : A.\phi)$

A *proposition* is a term of type Prop

The deduction rules are given by the usual natural deduction rules of introduction and elimination for implication and universal quantification

The system F_ω (Girard, 1970) is essentially a lambda calculus of proofs for minimal simple type theory

Minimal simple type theory

Leibniz equality $x =_A y$ is defined by $\forall f : A \to \mathsf{Prop}.\ \ f\ x \Rightarrow f\ y$

The *extensionality axioms* (Church 1940) are

$(\phi_1 \Leftrightarrow \phi_2) \Rightarrow \phi_1 =_{\mathsf{Prop}} \phi_2$

$(\forall x : A. f\ x =_B g\ x) \Rightarrow f =_{A \to B} g$

The axiom of *excluded midle* is

$\forall \phi : \mathsf{Prop}.\ \ \neg\neg\phi \Rightarrow \phi$

Minimal simple type theory

This system is *intensional* and *minimal*

There are translations from the extensional and classical systems into this more basic system:

the system with extensionality can be interpreted in the minimal system, a type becomes a type with an equivalence relation (R. Gandy, 1958)

the classical system can be interpreted in the minimal system by the so-called *negative* translation (Kolmogorov 1925, Gentzen, Gödel 1932)

Minimal simple type theory

The system is *impredicative*

It is illuminating to look at the interpretation in $\lambda*$

If instead we use a *predicative* universe for the type of propositions we cannot translate $\forall : (\mathsf{Prop} \to \mathsf{Prop}) \to \mathsf{Prop}$

This system has a finitary consistency proof by interpreting Prop as the finite set $\{0, 1\}$ (truth-table model)

Higher-order arithmetic

We can add a type Nat of natural numbers with constants $0 : \mathrm{Nat}$ and $S : \mathrm{Nat} \to \mathrm{Nat}$ and the usual Peano axioms

There is *no* finitary consistency proof any more

By the translations given above the system is as strong as its extensional classical version

Excluded Middle

We have two possible formulations of excluded-middle

$\forall A : \mathsf{U}.\neg\neg A \to A$ where $\neg A = A \to \mathsf{N}_0$

$\Pi A : \mathsf{U}.A + \neg A$

Exercise: Show that they are equivalent

$(\Pi A : \mathsf{U}.\neg\neg A \to A) \leftrightarrow (\Pi A : \mathsf{U}.A + \neg A)$

Excluded Middle

```
lem1 : {A : Set} -> neg (neg (A + neg A))
lem1 {A} = \ f -> f (Inr (\ p -> f (Inl p)))

lem2 : ((A : Set) -> neg (neg A) -> A) ->
       ((A : Set) -> A + neg A)
lem2 h A = h (A + neg A) lem1

lem3 : ((A : Set) -> A + neg A) ->
       ((A : Set) -> neg (neg A) -> A)
lem3 h A with h A
lem3 h A | Inl a = \ x -> a
lem3 h A | Inr b = \ x -> exit (x b)
```

Predicative type theory

We need to introduce an equality type Id as a *primitive* type

We can show one of Peano axiom

$$\neg(\mathsf{Id}\ 0\ (\mathsf{S}\ 0))$$

because we can define $F : \mathsf{Nat} \to \mathsf{U}$ such that $F\ 0$ is provable and $F\ (\mathsf{S}\ 0) = \bot$

Notice that we have used the universe U in this argument.

Question: can we prove $\neg(\mathsf{Id}\ 0\ (\mathsf{S}\ 0))$ in type theory without universes?

Predicative type theory

One can translate a *predicative* version of higher-order arithmetic, where one can quantify over natural numbers, functions, functionals, ... but *not* over propositions, predicates

The *intensional* axiom of choice is provable

$$(\Pi x : A.\Sigma y : B.R\ x\ y) \to \Sigma f : A \to B.\Pi x : A.R\ x\ (f\ x)$$

Aczel reducibility context

Assume that we can find an instance of the following context

$$B : \mathsf{U},\ \epsilon : \mathsf{U} \to B,\ T : B \to \mathsf{U}$$

$$ax : \Pi A : \mathsf{U}.\ \ A \leftrightarrow T\ (\epsilon\ A)$$

then we can use B as a *small* type of propositions and interpret higher-order arithmetic

Aczel reducibility context

Define

$$p_1 \Rightarrow p_2 = \epsilon(T\ p_1 \rightarrow T\ p_2)$$

$$\forall_A f = \epsilon(\Pi x : A.\ T\ (f\ x))$$

Then we can show

$$T\ (p_1 \Rightarrow p_2) \leftrightarrow (T\ p_1 \rightarrow T\ p_2))$$

$$T\ (\forall A\ f) \leftrightarrow \Pi x : A.\ T\ (f\ x))$$

Classical logic

If we add classical logic, for instance as the axiom

$$\Pi A : \mathsf{U}.\ \ A + \neg A$$

then we can find an instance of Aczel reducibility context

$$B = \mathsf{N}_2,\ T\ 0 = \mathsf{N}_0,\ T\ 1 = \mathsf{N}_1$$

Classical logic

```
T : N2 -> Set
T one = N1
T zero = N0
```

Classical logic

```
module class (axC : (A : Set) -> A + neg A) where

 funChoice : (A : Set) -> A + neg A -> N2
 funChoice A (Inl a) = one
 funChoice A (Inr b) = zero

 lemChoice : (A : Set) ->
             (x : A + neg A) -> A <-> T (funChoice A x)
 lemChoice A (Inl a) = (\ p -> unit)  , (\ q -> a)
 lemChoice A (Inr b) = b , exit
```

Classical logic

```
epsilon : Set -> N2
epsilon A = funChoice A (axC A)

prop : (A : Set) -> A <-> T (epsilon A)
prop A = lemChoice A (axC A)
```

Exercise: find a shorter derivation with the *with* construction

Classical logic

If we add classical logic to constructive type theory, it is possible to interpret impredicative quantification

Aczel reducibility context

If we strengthen the context

$$B : \mathsf{U},\ \epsilon : \mathsf{U} \to B,\ T : B \to \mathsf{U}$$

by asking $T\ (\epsilon\ X) = X$ then we can use B as a type of all types and we get an *inconsistent* system

Impredicative universe

Instead of having a type of all types, we can still keep the idea of propositions as types by having an *impredicative* universe Prop

An impredicative universe is a type Prop with a dependent type $T\ x\ [x : \text{Prop}]$ and an universal quantification $\forall : (A \to \text{Prop}) \to \text{Prop}$ with two maps

$$\text{in} : (\Pi x : A.T\ (f\ x)) \to T\ (\forall\ f), \quad \text{out} : T\ (\forall\ f) \to (\Pi x : A.T\ (f\ x))$$

such that $\text{out}\ (\text{in}\ u) = u \quad : \ \Pi x : A.T\ (f\ x)$

Impredicative universe

One can then interpret system F and even F_ω in this system

Also we can define Leibniz equality Id A at each type A

Proof-irrelevance states that for all x : Prop we can prove $\Pi p\ q : T\ x.$ Id $(T\ x)\ p\ q$

Proposition: *In impredicative type theory, classical logic implies proof irrelevance*

Impredicative universe

This is consistent with the fact that *classically* the interpretations of the propositions have to be "small"

If we have b : Prop such that $T\ b = \mathsf{N}_2$ then the system *negates* classical logic. Is it still consistent??

Lecture II will provide a model of such a system, hence a relative consistency proof

Asymmetry between universal/existential quantification

If we require $\exists : (A \to \text{Prop}) \to \text{Prop}$ with two maps

$$\text{in} : (\Sigma x : A.T\ (f\ x)) \to T\ (\exists\ f), \quad \text{out} : T\ (\exists\ f) \to (\Sigma x : A.T\ (f\ x))$$

with out $(\text{in}\ u) = u$ then we get an inconsistent system, since we have a retraction from Prop to the type $\exists x : \text{Prop}.a$ where $a : \text{Prop}$ is such that $T\ a$ is inhabited

A table of different systems

ZFC, ZF, topos theory, CZF, type theory, impredicative type theory w.r.t.

extensional axiom of choice

countable choice

impredicativity

extensionality

classical logic

Mathematics in type theory

We present two main ways of representing set theory in type theory

to interpret a set as a tree (or a pointed graph) with some bissimilarity relation as equality

to interpret a set as a type with an equivalence relation on this type

The W-type

We recall the type $Wx : A.B$ of well-founded trees given a type A and a family of types over it

Via Curry-Howard, this can be seen as a new *quantifier*

$\Pi x : \mathsf{N}_2.B$ is equivalent to $B(0) \wedge B(1)$

$\Sigma x : \mathsf{N}_2.B$ is equivalent to $B(0) + B(1)$

$Wx : \mathsf{N}_2.B$ is equivalent to $\neg(B(0) \wedge B(1))$

$Wx : A.B \to \neg(\Pi x : A.B)$

The W-type

```
data W (A : Set) (B : A -> Set) : Set where
  sup : (a : A) -> (B a -> W A B) -> W A B

lem : {A : Set} {B : A -> Set} ->
      ((x : A) -> B x) -> neg (W A B)
lem h (sup a f) = lem h (f (h a))
```

The W-type

Exercise: Show that

$W\ \mathsf{N}_2\ F \leftrightarrow \neg(F\ 0 \wedge F\ 1)$

if $F : \mathsf{N}_2 \to \mathsf{U}$

Sets as trees

This interpretation is due to Peter Aczel (1978)

The type of trees will then be $V = WX : \mathsf{U}.X$

Intuitively a tree $t = \sup\ I\ f$ is given by a small index set I and a family of subtrees $f : I \to V$

V is the type of *sets*

Equality is bissimulation which is defined inductively, it will be of type $V \to V \to \mathsf{U}$. Similarly membership will be of type $V \to V \to \mathsf{U}$

U is used both as a type of *propositions* and as the type of (small) index types for sets.

Sets as trees

We get an interpretation of a set theory known as CZF

In this set theory we don't have in general the power set axiom

The power set axiom holds iff Aczel reducibility context

$$B : \mathsf{U},\ \epsilon : \mathsf{U} \to B,\ T : B \to \mathsf{U}$$

$$ax : \Pi A : \mathsf{U}.\ \ A \leftrightarrow T\ (\epsilon\ A)$$

can be realised (which is very strong and not computational, as we have seen)

Sets as pointed graphs

A variation of this interpretation is due to Alexandre Miquel

This interpretation covers *non necessarily well-founded sets*

A point graph is simply a type A with an element $a : A$ and a relation $R : A \to A \to \mathsf{Prop}$

Equality is defined *coinductively* (which can be defined impredicatively as the union of all relations that are bissimulations)

An element of $X = (A, a, R)$ is (A, b, R) such that $R\ a\ b$ holds (we change the root of the graph)

Sets as pointed graphs

This gives the clearest way to get a paradox in $\lambda*$: if we apply this from $\lambda*$ we get a set theory in which we can interpret the general comprehension

$$\exists y.\forall x.(x \in y \leftrightarrow \phi(x))$$

which is contradictory by forming r such that

$$\forall x.(x \in r \leftrightarrow x \notin x)$$

Sets as types with equivalence relations

This interpretation comes from Bishop (1967) and it is remarkable how well his informal explanations in type theory

A *set* if a type A with an equivalence relation $=_A$

A set is defined when we describe how to construct its members ... and describe what it means for two members to be equal

Sets as types with equivalence relations

A *map* from $(A, =_A)$ to $(B, =_B)$ is a type theoretic function $f : A \to B$ such that $a_1 =_A a_2 \to f\ a_1 =_B f\ a_2$

If A and B are sets, then a function from A to B is a rule that assigns to each element of A an element of B and is extensional

We get in this way a *category* if we say that $f, g : (A, =_A) \to (B, =_B)$ are equal iff $\Pi a : A.\ \ f\ a =_B g\ a$ holds

One can explore the properties of this category dependent on the strength of the type theory

Sets as types with equivalence relations

Notice that this generalises Gandy's interpretation of *extensional* type theory in *intensional* type theory

$f : (A, =_A) \to (B, =_B)$ is *mono* iff $f\ a_1 =_B f\ a_2 \to a_1 =_A a_2$

$f : (A, =_A) \to (B, =_B)$ is *epi* iff $\Pi b : B.\Sigma a : A.\ b =_B f\ a$ (not so easy)

Some surprise: mono epi $\neq$ iso

It is also remarkable that Bishop defines a *subset* of X as a mono $Y \to X$

Sets as types with equivalence relations

The fact that any epi *splits* is a strong axiom, which is equivalent to the *extensional* version of the axiom of choice and which implies classical logic

Problem with defining model of lambda-calculus

General remark before starting the notion of model of type theory

If $\Gamma \vdash M : A$ and ρ an environment we try to define $[\![M]\!]_\rho$

For defining $[\![\lambda x.M]\!]_\rho$ it is not enough to know the set theoretic function

$$a \longmapsto [\![M]\!]_{\rho, x=a}$$

For instance, for domain models, we need to know also that this function is continuous.