Introduction to dependent type theory

CIRM, May 30

Goals of this presentation

Some history and motivations

Notations used in type theory

Main goal: the statement of main properties of equality type and the univalence axiom

First talk

$Propositions = Types$

Second talk

$Propositions = Types = Spaces$

History

Russell type theory 1903

Hilbert formulation of primitive recursion at higher types 1926

Brouwer intuitionistic logic, Kolmogorov's calculus of problems 1932

Gentzen natural deduction 1934

Church simplification of type theory, λ-calculus 1940

Gödel system T and Dialectica interpretation (1941) 1958

Curry discovery of the propositions-as-types principle 1958

History

Prawitz natural deduction 1965

Tait normalization proof for system T 1967

de Bruijn Automath 1967

Howard general formulation of proposition-as-types 1968

Scott Constructive Validity 1968 (strongly inspired by Automath)

Lawvere equality in hyperdocrtrines 1970

Girard system F and normalization proof 1970

History

Martin-Löf first system with a type of all types 1971 (inspired by Howard and Scott and Girard)

Girard's paradox 1971

Martin-Löf predicative system 1972, 1973 (formulation of the rules for identity)

Martin-Löf "extensional" type theory 1979

Bibliopolis book, 1984 (available on-line) still "extensional" version

Some notation

Application of a function to an argument $f\ a$

$f\ a_1\ a_2\ a_3$ for $((f\ a_1)\ a_2)\ a_3$

Notation used in combinatory logic and in functional programming (minimizes the use of parentheses)

Some notation

$A \to B \to C$ for $A \to (B \to C)$

λ-calculus notation for functions

$\lambda x.x^2 + 1$ denotes the function f such that $f\ a = a^2 + 1$

In general $(\lambda x.t)\ a$ is equal to $t(x = a)$.

The variable x is *bound* in $\lambda x.t$

Implication and exponentiation

$A \to B$ set of functions from A to B

$\lambda x.x$ is in $A \to A$

$\lambda x.\lambda y.x$ is in $A \to (B \to A)$

$\lambda f.\lambda x.f\ x\ x$ is in $(A \to A \to B) \to A \to B$

$\lambda g.\lambda f.\lambda x.g\ (f\ x)$ is in $(B \to C) \to (A \to B) \to (A \to C)$

We listed some of Hilbert's axioms for the implication

Implication and exponentiation

$\lambda x.\lambda f.f\ x$ is in $A \to (A \to X) \to X$

The false proposition $\bot$ corresponds to the empty set $\emptyset$

$\bot \to A$ corresponds to $\emptyset \to A$

$(A \to \emptyset) \to \emptyset$ is empty if A is empty and has one element if A is nonempty

Propositions as Sets?

$A \times B$ corresponds to $A \wedge B$

$A + B$ corresponds to $A \vee B$

$\lambda f.\lambda z.f\ z.1\ z.2$ is in $(A \to B \to X) \to A \times B \to X$

$\lambda f.\lambda x.\lambda y.f\ (x, y)$ is in $(A \times B \to X) \to A \to B \to X$

Propositions as Sets?

$(A \wedge (A \to B)) \to B$ corresponds to

$(A \times (A \to B)) \to B$ which is inhabited by $\lambda z.z.2\ (z.1)$

$\lambda f.\lambda g.\lambda x.g\ (f\ x)$ inhabits $(A \to B) \to \neg B \to \neg A$

$(\neg B \to \neg A) \to A \to B$ does not have any uniform inhabitant

Dependent products and sums

Family of sets A_i over a set I

$\prod_{i \in I} A_i$ set of sections $(a_i)_{i \in I}$ or f such that $f\ i \in B_i$ for all i

$\sum_{i \in I} A_i$ set of pairs (i, a) with $i \in I$ and $a \in A_i$

If A_i is a constant family $A_i = A$ for all i this reduces to $I \to A$ and $I \times A$ respectively

Dependent products and sums

$\lambda f.\lambda i.(f.1\ i, f.2\ i)$ is in $(\prod_{i \in I} B_i \times \prod_{i \in I} C_i) \to \prod_{i \in I} B_i \times C_i$

$\lambda g.(\lambda i.(g\ i).1, \lambda i.(g\ i).2)$ is in $(\prod_{i \in I} B_i \times C_i) \to (\prod_{i \in I} B_i \times \prod_{i \in I} C_i)$

This corresponds to the logical equivalence

$(\forall x.B(x) \wedge C(x)) \leftrightarrow (\forall x.B(x) \wedge \forall x.C(x))$

The axiom of choice

$\lambda g.(\lambda i.(g\ i).1, \lambda i.(g\ i).2)$ is in

$$(\prod_{i\in I} \sum_{j\in J} A_{i,j}) \to \sum_{f\in I\to J} \prod_{i\in I} A_{i,f\ i}$$

$\sum_{i\in I} B_i$ corresponds to "strong existential": if c is in $\sum_{i\in I} B_i$ then $c.1$ is in I and $c.2$ is in $B_{c.1}$

reminiscent of $(\exists x)P(x) \to P(\epsilon(P))$ but the $\epsilon(P)$ depends now on the *proof* of $(\exists x)P(x)$

Primitive recursion

$\mathbb{N}$ is generated by 0 and $x + 1$

If a is in X and g is in $\mathbb{N} \to X \to X$ we can define f in $\mathbb{N} \to X$ by the equations

$$f\ 0 = a \qquad\qquad f\ (n + 1) = g\ n\ (f\ n)$$

Hilbert 1926

Primitive recursion

If B_n family of types over $n \in \mathbb{N}$ and a is in B_0 and g is in $\prod_{n \in \mathbb{N}} (B_n \to B_{n+1})$

we can define f in $\prod_{n \in \mathbb{N}} B_n$ by the equations

$$f\ 0 = a \qquad\qquad f\ (n+1) = g\ n\ (f\ n)$$

This corresponds to the principle of induction in Peano arithmetic

Intuitionistic theory of types

So far, we were looking at examples in set theory

From now on, we describe type theory as a formal system (not necessarily based on type theory; indeed the univalence axiom is contradictory with set theory)

We consider mathematical objects

Each object comes with its type $a : A$

A *proposition* is defined by prescribing how we are allowed to prove it

We represent each proposition as a type (the type of its proofs)

Intuitionistic theory of types

If A is a type and $B(x)$ is a method which to an arbitrary object of type A associates a type $B(a)$ then we can form the type $\prod_{x:A} B(x)$

If b is of type $\prod_{x:A} B(x)$ and $a : A$ then $b\ a$ is of type $B(a)$

If $t(x)$ is of type $B(x)$ for $x : A$ then $\lambda x.t(x)$ is of type $\prod_{x:A} B(x)$

If we have $B(x) = B$ then $\prod_{x:A} B$ is written $A \to B$

Intuitionistic theory of types

If A is a type and $B(x)$ is a method which to an arbitrary object of type A associates a type $B(a)$ then we can form the type $\sum_{x:A} B(x)$

If $a : A$ and $b : B(a)$ then (a, b) is of type $\sum_{x:A} B(x)$

If $c : \sum_{x:A} B(x)$ then $c.1 : A$ and $c.2 : B(c.1)$

If we have $B(x) = B$ then $\sum_{x:A} B$ is written $A \times B$

Intuitionistic theory of types

If $B(x)$ represents a proposition then $\sum_{x:A} B(x)$ can be thought of as $(\exists x : A)B(x)$

But it represents also $\{x : A \mid B(x)\}$

Type of real numbers

$$\sum_{x:N\to Q} \prod_{m:N} \prod_{n:N} |x\ (m+n) - x\ m| \leqslant 2^{-m}$$

Disjoint union

If A and B are types we can form the type $A + B$

If $a : A$ then $\mathsf{inl}\ a : A + B$ and if $b : B$ then $\mathsf{inr}\ b : A + B$

If we have a family of type $C(z)$ over $z : A + B$

we have $u : C(\mathsf{inl}\ x)$ for $x : A$ and $v : C(\mathsf{inr}\ y)$ for $y : B$

then we can define $f : \prod_{z:A+B} C(z)$ by the equations

$f\ (\mathsf{inl}\ x) = u$ $\qquad$ $f\ (\mathsf{inr}\ y) = v$

Disjoint union

The language of the theory is richer than the language of traditional systems in permitting proofs to appear as parts of the propositions so that the propositions can express properties of proofs (and not only individuals, like in first-order logic). This makes it possible to strengthen the axioms for existence, disjunction, absurdity and identity.

Disjoint union

Usual rule for disjunction

introduction rules

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

elimination rule

$$\frac{C\ [A] \qquad C\ [B] \qquad A \vee B}{C}$$

Natural numbers

N is a type of *constructors* $0 : N$ and $\mathsf{S}\ x : N$ if $x : N$

Let $C(n)$ be a function which to an arbitrary natural number $n : N$ assigns a type

Given $d : C(0)$ and $e : \prod_{n:N} (C(n) \to C(\mathsf{S}\ n))$

we may introduce a function $f : \prod_{n:N} C(n)$ by the equations

$f\ 0 = d \qquad f\ (\mathsf{S}\ n) = e\ n\ (f\ n)$

Inductive definitions

We can introduce the type N_1 with only one constructor $0 : N_1$

If $C(x)$ type for $x : N_1$ and we have $a : C(0)$

we can introduce $f : \prod_{x:N_1} C(x)$ by the equation

$f\ 0 = a$

Inductive definitions

We can introduce the type N_2 with two constructors $0 : N_2$ and $1 : N_2$

If $C(x)$ type for $x : N_2$ and we have $a_0 : C(0)$ and $a_1 : C(1)$

we can introduce $f : \prod_{x:N_2} C(x)$ by the equations

$f\ 0 = a_0$ $\qquad$ $f\ 1 = a_1$

Inductive definitions

We introduce the type N_0 with *no* constructor

If $C(x)$ type for $x : N_0$

we have $f : \prod_{x:N_0} C(x)$

A particular case is $N_0 \to X$ for any type X

Inductive definitions

All these constructions follow the same pattern

constructors correspond to primitive operations and to introduction rules in natural deduction

defined functions correspond to elimination rules that are justified by

computation rules which correspond to normalization in natural deduction

The process of following these rules correspond to computations of functions and to the process of understanding a proof by eliminating lemmas

Inductive definitions

N, N_2, N_1, N_0 and $A+B$ are examples of type introduced by *ordinary inductive definition*

Hilbert 1926 considers the type Ord of *ordinal numbers*

$0 : Ord$

if $x : Ord$ then $\mathsf{S}\ x : Ord$

if $u : N \to Ord$ then its limit $L\ u : Ord$

Inductive definitions

If $c : C(0)$ and $g : \prod_{x:Ord} (C(x) \to C(\mathsf{S}\ x))$ and

$$h : \prod_{u:N\to Ord} (\prod_{n:N} C\ (u\ n)) \to C\ (L\ u)$$

we may introduce $f : \prod_{x:Ord} C(x)$ by the *recursion* schema

$$f\ 0 = c \qquad f\ (\mathsf{S}\ x) = g\ x(f\ x) \qquad f\ (L\ u) = h\ u\ (f \circ u)$$

where $(f \circ u)\ x = f\ (u\ x)$

Inductive definitions

Thinking of $C(x)$ as a proposition, f is a proof of the universal proposition $(\Pi x \in Ord)C(x)$ which we get by applying the principle of *transfinite induction* over the second number class ordinals.

Type theory

In the formal theory the abstract entities (natural numbers, ordinals, functions, types, and so on) become represented by certain symbol configurations, called terms, *and the definitional schema, read from the left to the right, become mechanical* reduction rules *for these symbol configurations.*

Type theory *effectuates the computerization of abstract intuitionistic mathematics that above all Bishop has asked for*

It provides a framework in which we can express *conceptual* mathematics in a *computational* way.

Type theory

A term is *normal* if it cannot be further reduced

The closed normal term of type N_2 are exactly 0 and 1

The closed normal term of type N have the form $0,\ \mathsf{S}\ 0,\ \mathsf{S}\ (\mathsf{S}\ 0), \ldots$

There is no normal closed term of type N_0

Hence if we have normalization (and preservation of types by reduction) there is *no* closed term of type N_0

Since $\bot = N_0$ this expresses the *logical consistency* of type theory

Context

Notion of *context* (introduced by Automath, N. de Bruijn)

$x_1 : A_1,\ x_2 : A_2(x_1),\ x_3 : A_3(x_1, x_2), \ldots$

"let x be a natural number, assume that $\varphi(x)$ holds for x, and let y be a rational number, $\ldots$"

$x : N,\ h : \varphi(x),\ y : Q, \ldots$

Compared to the usual mathematical notation, notice that we have an explicit name for the hypothesis that $\varphi(x)$ holds.

$a_1, \ldots, a_n$ fits $x_1 : A_1, \ldots, x_n : A_n$ iff

$a_1 : A_1,\ a_2 : A_2(a_1),\ \ldots,\ a_n : A_n(a_1, \ldots, a_{n-1})$

Context

$\Gamma, \Delta, \ldots$ for context

Hypothetical judgements $\Delta \vdash a : A$

Interpretation $\Delta \to \Gamma$ if $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

sequence $a_1, \ldots, a_n$ such that

$$\Delta \vdash a_1 : A_1,\ \Delta \vdash a_2 : A_2(a_1),\ \ \ldots,\ \Delta \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

The type of types

The idea of the type of types is forced upon us by accepting simultaneously each of the following three principles.

First, quantification over propositions as in second order logic.

Second, Russell's doctrine of types according to which the ranges of significance of propositional functions form types so that, in particular, it is only meaningful to quantify over all objects of a certain type.

Third, the identification of propositions and types.

Suppose namely that quantification over propositions is meaningful. Then by the doctrine of types, the propositions must form a type. But, if propositions and types are identified, then this type is at the same time the type of types.

The type of types

It is natural to introduce a type U of all types

In particular $U : U$

The type of types introduces a strong kind of selfreference which, as pointed out by Gödel 1964, transcends the cumulative hierarchy notion of set and may seem to verge on the paradoxes, but which is actually being used in category theory, notably, in the construction of the category of all categories

The type of types

We can define types by recursion, for instance $T : N \to U$

$T\ 0 = N \qquad T\ (n+1) = (T\ n) \to N$

but also $Eq : N \to N \to U$

$Eq\ 0\ 0 = N_1 \qquad Eq\ (n+1)\ (m+1) = Eq\ n\ m$

$Eq\ 0\ (m+1) = Eq\ (n+1)\ 0 = N_0$

Girard's paradox

It does not seem possible to translate directly Russell's paradox

We can form $\sum_{X:U} B(X)$ but $B(X)$ cannot express that X is not of type X

The "judgement" $a : A$ is not a proposition that can be negated

Girard's paradox

We can however translate Burali-Forti's paradox

We define we a relation $R : A \to A \to U$ is transitive

$$P\ A\ R = \prod_{x\ y\ z:A} R\ x\ y \to R\ y\ z \to R\ x\ z$$

and well-founded

$$Q\ A\ R = \prod_{f:N\to A} \neg\ (\prod_{n:N} R\ (f\ (n+1))\ (f\ n))$$

where $\neg\ X = X \to \bot = X \to N_0$

Girard's paradox

We can form the type of all well-founded relations

$$V = \sum_{A:U} \sum_{R:A\to A\to U} (P\ A\ R) \times (Q\ A\ R)$$

An element in V is a tuple A, R, p, q where $p : P\ A\ R$ and $q : Q\ A\ R$

We define then $R_V : V \to V \to U$ with $p_V : P\ V\ R_V$ and $q_V : Q\ V\ R_V$ so that we have an element

$$v_0 : V = (V, R_V, p_V, q_V)$$

Girard's paradox

We show

$$\prod_{x:V} R_V\ x\ v_0$$

and in particular

$$R_V\ v_0\ v_0$$

using q_V (proof that R_V is well-founded), we get a closed term of type N_0

This term is not normalizable

(intuitively, we "loop" when we try to understand this proof)

Universe

The incoherence of the idea of a type of all types whatsoever made it necessary to distinguish, like in category theory, between small and large types. Thus the universe U appears, not as the type of all types, but as the type of small types, whereas U itself and all types which are built up from it are large. This makes the types wellfounded and the theory predicative.

The situation is reminiscent of the situation in set theory after Russell's paradox

New information: formally one can prove normalization of the inconsistent system in the same way as one proves normalization of consistent system (this casts some doubt on normalization proofs...)

Universe

This is reminiscent of Grothendieck's notion of universe

If $A : U$ and $B : U$ for $x : A$ then we have $\prod_{x:A} B : U$

If $A : U$ and $B : U$ for $x : A$ then we have $\sum_{x:A} B : U$

$N_0,\ N_1,\ N_2, N$ are all of types U

We can define small types by recursion

Universes

We can prove normalization (in ZFC or second-order arithmetic)

The system we obtain is *predicative*. The "proof-theoretic strength" is weaker than second-order arithmetic (weaker than Π^1_2 comprehension)

$\prod_{X:U} X + \neg X$ is not provable

$\prod_{X:U} (\neg\neg X) \to X$ is not provable

$\prod_{X:U} \neg\neg(X + \neg X)$ is provable

Universes

Define $T : N_2 \to U$ by $T\ 0 = N_0 \quad T\ 1 = N_1$

then we can prove

$$\prod_{X:U} ((X + \neg X) \quad \leftrightarrow \quad \sum_{b:N_2} (X \leftrightarrow T\ b))$$

Church's formulation of type theory

Simplification of Russell's theory of types

Tarski, Gödel presentation of type theory: if $\alpha_1, \ldots, \alpha_n$ are types ($n \geqslant 1$) then $(\alpha_1, \ldots, \alpha_n)$ is the type of relations

Church introduces a type of proposition o, a type of individuals and function types $A \to B$

For instance $o \to o$ is the type of the operation of negation

But we have also the type of functions $\iota \to \iota$

Church's formulation of type theory

We have the usual connectives on propositions

$p \to q : o$ for the implication if $p\ q : o$

quantifiers at any type $\forall x : A.\varphi :\ o$ if $\varphi : o\ [x : A]$

Church's formulation

Uses λ-calculus to represent terms (implicit in *Principia Mathematica*)

If $f : A \to B$ and $a : A$ then $f\ a : B$ the application of the function f to the argument a

If $t : B\ [x : A]$ then $\lambda x.t : A \to B$

The terms of type o are the propositions

Usual connectives and (classical) logical rules

Type theory

We can try to interpret higher-order logic in type theory

o is interpreted as U

$A \to B$ used both for function types and logical implication

$\prod_{x:A} B$ for $\forall x : A.B$

However $\forall x : o.B$ is of type o while $\prod_{x:U} B$ is *not* of type U if $B : U$ for $x : U$

Impredicative Type theory

We can interpret higher-order logic if we have $U : U$

What we need is weaker (impredicative type theory)

$\prod_{x:A} B : U$ if $B : U$ for $x : A$ *without* any restriction on A

This system has the normalization property

We cannot think of types as sets in ZFC

One can prove $\neg \left(\prod_{X:U} X + \neg X\right)$ in this system

Impredicative Type theory

In impredicative type theory we can define the equality following Leibnitz/Principia Mathematica

$$\prod_{X:A\to U} X\ a_0 \to X\ a_1$$

expresses that the elements $a_0\ a_1 : A$ satisfy the same properties

$$\lambda a_0.\lambda a_1.\ \prod_{X:A\to U} X\ a_0 \to X\ a_1$$

is of type $A \to A \to U$

The W type

$\prod_{x:A} B$ and $\sum_{x:A} B$

$(\Pi x : A)B$ and $(\Sigma x : A)B$ universal and existential quantifiers

New quantifier $(W\ x : A)B$

Constructor $\sup\ a\ f : (W\ x : A)B$

if $a : A$ and $f : B(a) \to (W\ x : A)B$

The W type

If $d : \prod_{x:A} \prod_{f:B(x)\to(W\ x:A)B} (\prod_{y:B(x)} C\ (f\ y)) \to C\ (\mathrm{sup}\ x\ f)$

we may introduce $g : \prod_{t:(W\ x:A)B} C(t)$ by the defining equation

$g\ (\mathrm{sup}\ a\ f) = d\ a\ f\ (\lambda y.g\ (f\ y))$

The W type

This can be seen as a new *quantifier*

$(\Pi x : N_2)B$ is equivalent to $B(0) \wedge B(1)$

$(\Sigma x : N_2)B$ is equivalent to $B(0) + B(1)$

$(Wx : N_2)B$ is equivalent to $\neg(B(0) \wedge B(1))$

$(W\ x : A)B \to \neg(\Pi x : A)B$

$g : (W\ x : A)B \to ((\Pi x : A)B) \to N_0$ defined by

$g\ (\sup\ a\ f)\ h = g\ (f\ (h\ a))\ h$

Type theory and set theory

Consider $V = (W\ X : U)X$

An element of $\alpha : V$ can be thought of as a small type $\alpha_0 : U$ with a function $\alpha_1 : \alpha_0 \to V$

We define inductively the equality on V (bissimulation)

$\sup\ \alpha_0\ \alpha_1 =_V \sup\ \beta_0\ \beta_1$ is

$$(\prod_{x:\alpha_0} \sum_{y:\beta_0} \alpha_1\ x =_V \beta_1\ y) \times (\prod_{y:\beta_0} \sum_{x:\alpha_0} \alpha_1\ x =_V \beta_1\ y)$$

Type theory and set theory

We can then define a membership relation

$\alpha \ \epsilon \ \beta$ is defined as $\sum_{y:\beta_0} \alpha =_V \beta_1 \ y$

Most of the axioms of set theory are validated by this interpretation

For instance $\exists! x. \forall y. \neg (y \ \epsilon x)$

or $\forall x \ y. \exists z. \forall t. \ t \ \epsilon \ z \leftrightarrow (t = x \vee t = y)$

This interpretation justifies CZF (constructive version of ZF)

If we have $U : U$ this interpretation justifies naive set theory (and we can reproduce Russell's paradox)

Axiom of choice

$$(\prod_{x:A} \sum_{y:B} R\ x\ y) \to \sum_{f:A\to B} \prod_{x:A} R\ x\ (f\ x)$$

is inhabited by $\lambda g.(\lambda x.(g\ x).1, \lambda x.(g\ x).2)$

Type-checking uses that $(\lambda x.(g\ x).1)\ a$ and $(g\ a).1$ are identical

This expresses well the fact that in constructive mathematics, the axiom of choice holds by the very meaning of existence (Bishop)

This is quite different from the axiom of choice in topos theory

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