Intuitionistic Type Theory
Lecture 3

Peter Dybjer

Chalmers tekniska högskola, Göteborg

Summer School on Types, Sets and Constructions
Hausdorff Research Institute for Mathematics
Bonn, 3 - 9 May, 2018

Plan of lectures on Intuitionistic Type Theory

1. formal system of 1972
2. meaning explanations in "Constructive Mathematics and Computer Programming" 1979/82; identity types
3. formal system of 1986 as a basis for
   - the proof assistant / dependently typed programming language Agda
   - a general theory of inductive and inductive-recursive definitions

Another interesting topic:

- What is dependent type theory? Initial categories with families built by categorical combinators for dependent types.

The LF version of Intuitionistic Type Theory 1986

A two level theory:

- the *theory of types* (the logical framework); only two type formers
  - $(x : A) \to B$ (like $\Pi x : A.B$ but serving a different role)
  - Set (like U but serving a different role)
  - (We will also need $(x : A) \times B$, like $\Sigma x : A.B$)
- the *theory of sets*; contains the rules for the type formers.
  - formation, introduction, and elimination-rules are represented by constants with their types
  - equality rules are represented by equations

Variable binding and substitution taken care of on the framework level.
Cf Edinburgh Logical Framework (Harper, Honsell, Plotkin 1987).

Agda

The proof assistant Agda is based on (a modified version of) Martin-Löf's logical framework. It contains numerous programming language features:

- modules
- records
- a universe hierarchy $\mathrm{Set} : \mathrm{Set}_1 : \mathrm{Set}_2 : \cdots$
- data type declarations (of inductive and inductive-recursive sets)
- pattern matching and termination checking (as a more flexible alternative to elimination rules)
- implicit arguments
- syntactic sugar including syntax highlighting with colours, unicode, mixfix operations, precedence declarations, etc.
- an emacs-interface where programs/proofs can be written by gradual refinement of (typed checked) partial terms

Rules for natural numbers in Agda

```
module N-rules where

data ℕ : Set where
  O : ℕ
  s : ℕ → ℕ

R : {C : ℕ → Set}
  → C O
  → ((n : ℕ) → C n → C (s n))
  → (c : ℕ) → C c
R d e O     = d
R d e (s n) = e n (R d e n)
```

Rules for the identity set in Agda

```
module I-rules where

data I (A : Set) (a : A) : A → Set where
  r : I A a a

J : {A : Set} → {a : A}
  → (C : (y : A) → I A a y → Set)
  → C a r
  → (b : A) → (c : I A a b) → C b c
J C d b r = d
```

Intuitionistic Type Theory 1986 have decidable judgments

The following rules are removed:

Identity reflection

$$\frac{\Gamma \vdash c : I(A, a, a')}{\Gamma \vdash a = a' : A}$$

Uniqueness of identity proofs:

$$\frac{\Gamma \vdash c : I(A, a, a')}{\Gamma \vdash c = r : I(A, a, a')}$$

Decidability of the judgments is restorted and Agda's type-checking algorithm decides the judgment

$$\vdash a : A$$

Intuitionistic Type Theory as a Theory of Inductive Definitions

Martin-Löf 1972:

> *The type N is just the prime example of a type introduced by an ordinary inductive definition. However, it seems preferable to treat this special case rather than to give a necessarily much more complicated general formulation which would include* $(\Sigma \in A)B(x), A + B, N_n$ *and N as special cases. See Martin-Löf 1971 [26] for a general formulation of inductive definitions in the language of ordinary first order predicate logic.*

Iterated inductive definitions in predicate logic

What is an iterated inductive definition of predicate symbols $P, Q, \ldots, R$, *in general* in natural deduction? What are their introduction, elimination, and "contraction" rules?
Ordinary production:

$$\frac{Q(q(x)) \quad \cdots \quad R(r(x))}{P(p(x))}$$

Generalized productions:

$$\frac{H(x) \supset Q(q(x))}{P(p(x))} \qquad \frac{\forall y.Q(q(x,y))}{P(p(x))}$$

There are restrictions on the level of predicates. H is an arbitrary formula of lower level than P.

Inductive families

General form of introduction rule for an *inductive family*.

$$\frac{\cdots \qquad \Gamma \vdash a : A \qquad \cdots \qquad \Gamma, y : H \vdash b : P(q) \qquad \cdots}{\Gamma \vdash c(\ldots, a, \ldots, y.b, \ldots) : P(p)}$$

"Inductive predicates with proof objects". *Inductive types* are special cases.

- P is the type constructor for the inductive family.
- c is a term constructor for P.
- $\Gamma \vdash a : A$ is a *side condition* or *non-inductive premise*. There can be several. A must be defined before ("have lower level").
- $\Gamma, y : H \vdash b : P(q)$ is an *inductive premise*. There can be several. H must be defined before.

Lists and vectors

An inductive type:

$$\Gamma \vdash \mathit{nil} : \mathit{List}(A) \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash \mathit{as} : \mathit{List}(A)}{\Gamma \vdash \mathit{cons}(a, \mathit{as}) : \mathit{List}(A)}$$

An inductive family:

$$\Gamma \vdash \mathit{nil} : \mathit{Vect}(A, 0) \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash \mathit{as} : \mathit{Vect}(A, n)}{\Gamma \vdash \mathit{cons}(a, \mathit{as}) : \mathit{Vect}(A, s(n))}$$

A is a parameter.

Generalized inductive definitions

W-formation.

$$\frac{\Gamma \vdash A \qquad \Gamma, x : A \vdash B}{\Gamma \vdash Wx : A.B}$$

W-introduction.

$$\frac{\Gamma \vdash a : A \qquad \Gamma, y : B[x := a] \vdash b : Wx : A.B}{\Gamma \vdash sup(a, y.b) : Wx : A.B}$$

Martin-Löf 1979.

Iterative sets

Aczel's constructive cumulative hierarchy:

$$V = Wx : U.x.$$

for a family of iterative sets $x : A \vdash M : V$, we form

$$\{M \,|\, x : A\} = sup(A, x.M) : V$$

W-types and schema for inductive definitions

W-introduction ($W = Wx : A.B$).

$$\frac{\Gamma \vdash a : A \qquad \Gamma, y : B[x := a] \vdash b : W}{\Gamma \vdash sup(a, y.b) : W}$$

General form of an introduction rule for an inductive type W

$$\frac{\cdots \qquad \Gamma \vdash a : A \qquad \cdots \qquad \Gamma, y : B \vdash b : W \qquad \cdots}{\Gamma \vdash sup(\ldots, a, \ldots, y.b, \ldots) : W}$$

The general schema can be reduced to W-types modulo some extensional isomorphisms which are not valid in the intensional theory.

Universe à la Tarski (N, Π, U-fragment)

$$\frac{\Gamma \vdash a : U \qquad \Gamma, x : T(a) \vdash b : U}{\Gamma \vdash \pi(a, x.b) : U} \qquad \Gamma \vdash \hat{n} : U$$

$$\begin{aligned} T(\pi(a, x.b)) &= \Pi x : T(a).T(b(x)) \\ T(\hat{n}) &= N \end{aligned}$$

Universe à la Tarski (N, Π, U-fragment)

```
module UT-rules where
open import N-rules
open import Pi-rules

mutual
  data U : Set where
    π : (a : U) → (T a → U) → U
    n : U

  T : U → Set
  T (π a b) = Π (T a) (λ x → T (b x))
  T n       = ℕ
```

Examples of inductive-recursive definitions in type theory

Constructive higher infinite:

- Palmgren's next universe operators and super universe
- Rathjen, Griffor, and Palmgren's universe for Mahlo π-numbers
- Setzer's Mahlo universe
- Palmgren's universe hierarchies

Intuitionistic model theory:

- modelling types and terms in Frege structures (Aczel)
- computable types and terms for a normalization proof (Martin-Löf)

Small inductive-recursive defintions:

- "fresh lists", etc

General schema for inductive-recursive types

Given a type C, we define by simultaneous induction-recursion

$$\begin{aligned} U &: Set \\ T &: U \to C \end{aligned}$$

The schema is as before

$$\frac{\cdots \qquad \Gamma \vdash a : A \qquad \cdots \qquad \Gamma, y : B \vdash b : U \qquad \cdots}{\Gamma \vdash sup(\ldots, a, \ldots, y.b, \ldots) : U}$$

with

$$T(sup(\ldots, a, \ldots, y.b, \ldots)) = \ldots$$

where side conditions and inductive premises can come in any order. The A and the B may now depend on T applied to previously constructed elements of U. How to make this precise?

Finite axiomatization of inductive types

A universe Sig of codes Σ for inductive types U_Σ

$$\begin{array}{lcl} \varepsilon & : & \mathrm{Sig} \\ \sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}) \to \mathrm{Sig} \\ \rho & : & \mathrm{Set} \to \mathrm{Sig} \to \mathrm{Sig} \end{array}$$

The associated functor F_Σ

$$\begin{array}{rcl} F_\varepsilon\, X & = & 1 \\ F_{\sigma\, A\, \Sigma}\, X & = & (x : A) \times F_{\Sigma\, x}\, X \\ F_{\rho\, A\, \Sigma}\, X & = & (A \to X) \times F_\Sigma\, X \end{array}$$

The rules for U_Σ are obtained from the initial F_Σ-algebra diagram.

Some codes

Some defined codes

$$\begin{aligned} id &= \rho\, 1\, \varepsilon \\ \Sigma + \Sigma' &= \sigma\, 2\, (\lambda x.\mathit{if}\ x\ \mathit{then}\ \Sigma\ \mathit{else}\ \Sigma') \\ \Sigma_N &= \varepsilon + id \\ \Sigma_{WAB} &= \sigma\, A\, (\lambda x.\rho\, (B\, x)\, \varepsilon) \end{aligned}$$

with their endofunctors

$$\begin{aligned} F_{id}\, X &\cong X \\ F_{\Sigma+\Sigma'}\, X &\cong F_\Sigma\, X + F_{\Sigma'}\, X \\ F_{\Sigma_N}\, X &\cong 1 + X \\ F_{\Sigma_{WAB}}\, X &\cong (x : A) \times (B\, x \to X) \end{aligned}$$

Finite axiomatization of inductive-recursive types

An inductive-recursive definition is obtained by reflecting an operation

$$\phi \quad : \quad G_{C,\Sigma} \to C$$

as a constructor

$$c_\Sigma \quad : \quad G^c_{C,\Sigma}\, U_\Sigma\, T_\Sigma \to U_\Sigma$$

for an inductively defined set with a recursively defined decoding

$$\begin{aligned} U_\Sigma &\quad : \quad \mathrm{Set} \\ T_\Sigma &\quad : \quad U_\Sigma \to C \end{aligned}$$

where $G_{C,\Sigma}$ is a "good" domain of definition for all $\Sigma : \mathrm{Sig}_C$.

Reflecting Set-valued operations

An inductive-recursive definition is obtained by reflecting an operation

$$\phi \quad : \quad G_\Sigma \to \mathrm{Set}$$

as a constructor

$$c_\Sigma \quad : \quad G^c_\Sigma \, U_\Sigma \, T_\Sigma \to U_\Sigma$$

for an inductively defined set with a recursively defined decoding

$$\begin{aligned} U_\Sigma &\quad : \quad \mathrm{Set} \\ T_\Sigma &\quad : \quad U_\Sigma \to \mathrm{Set} \end{aligned}$$

where G_Σ is a "good" domain of definition for all $\Sigma : \mathrm{Sig}_{\mathrm{Set}}$.

Finite axiomatization of inductive-recursive types

$\mathrm{Sig}_{\mathrm{Set}}$ is a type of "signatures" for inductive-recursive definitions reflecting operations ϕ with codomain Set

$$\begin{array}{lcl} \varepsilon & : & \mathrm{Sig}_{\mathrm{Set}} \\ \sigma & : & (A : \mathrm{Set}) \to (A \to \mathrm{Sig}_{\mathrm{Set}}) \to \mathrm{Sig}_{\mathrm{Set}} \\ \rho & : & (A : \mathrm{Set}) \to ((A \to \mathrm{Set}) \to \mathrm{Sig}_{\mathrm{Set}}) \to \mathrm{Sig}_{\mathrm{Set}} \end{array}$$

generating the good domains of definition for ϕ

$$\begin{array}{rcl} G_{\varepsilon} & = & 1 \\ G_{\sigma\, A\, \Sigma} & = & (x : A) \times G_{\Sigma\, x} \\ G_{\rho\, A\, \Sigma} & = & (f : A \to \mathrm{Set}) \times G_{\Sigma\, f} \end{array}$$

Code for universe closed under Π

The code

$$\Sigma_\Pi = \rho\,1\,(\lambda A.\rho\,(A0)\,(\lambda B.\varepsilon)) : \mathrm{Sig}_{\mathrm{Set}}$$

generates the domain of Π (uncurried)

$$G_{\Sigma_\Pi} : (A : 1 \to \mathrm{Set}) \times ((A0) \to \mathrm{Set}) \times 1 \cong (A : \mathrm{Set}) \times (A \to \mathrm{Set})$$

Finite axiomatization of inductive-recursive types

The introduction rule is

$$c_\Sigma \quad : \quad G^c_\Sigma\, U_\Sigma\, T_\Sigma \to U_\Sigma$$

(Note the dependence on both U_Σ and T_Σ). We have

$$\begin{aligned} G^c_\varepsilon\, U\, T &= 1 \\ G^c_{\sigma\, A\, \Sigma}\, U\, T &= (x : A) \times G^c_{\Sigma\, x}\, U\, T \\ G^c_{\rho\, A\, \Sigma}\, U\, T &= (f : A \to U) \times G^c_{\Sigma\, (T \circ f)}\, U\, T \end{aligned}$$

which generate the domain of π (the code for Π):

$$\begin{aligned} G^c_{\Sigma_\Pi}\, U_{\Sigma_\Pi}\, T_{\Sigma_\Pi} \quad : \quad & (f : 1 \to U_{\Sigma_\Pi}) \times (T_{\Sigma_\Pi}\, (f\, 0) \to U_{\Sigma_\Pi}) \times 1 \\ \cong \quad & (a : U_{\Sigma_\Pi}) \times (T_{\Sigma_\Pi}\, a \to U_{\Sigma_\Pi}) \end{aligned}$$

Finite axiomatization of inductive-recursive types

Remains:

- Equality rule for $T_{\Sigma\Pi}$.
- Reflection of C-valued operations. Easy.
- Indexed induction-recursion.
- Universe-elimination for inductive-recursive types in general.

Three views of Intuitionistic Type Theory

- Intuitionistic Type Theory with a fixed collection of basic type formers (e g $\emptyset, 1, +, \Sigma, \Pi, N, U$)
- Intuitionistic Type Theory as a general theory of inductive and inductive-recursive definitions (PD, A Setzer)
  - A finite axiomatization of inductive-recursive definitions, TLCA 1999
  - Induction-recursion and initial algebras, APAL, 2003
  - Indexed induction-recursion, JLAP, 2006
- Intuitionistic Type Theory as an open theory, cf Agda, e g add Setzer's autonomous Mahlo universe, and Π_3-reflecting universe.

Formation and introduction rules fit the pattern of meaning explanations.

Uses of the word "set" in Intuitionistic Type Theory

- set as a synonym for type (Bibliopolis)
- set as small type (logical framework formulation)
- set as setoid (type + equivalence relation, Bishop set)
- set as iterative set in $(V, =_V)$, Aczel's model of CZF
- set as hset in homotopy type theory