Martin-Löf's Meaning Explanations for Intuitionistic Type Theory

Peter Dybjer

Chalmers tekniska högskola, Göteborg

Initial Types Club
3 February 2022

Constructive Mathematics and Computer Programming

Per Martin-Löf, in Logic, Methodology and Philosophy of Science, Hannover 1979.

> *(p 15) In explaining what a judgment of the above four forms means, I shall first limit myself to assumption free judgments. Once it has been explained what meanings they carry, the explanations can readily be extended so as to cover hypothetical judgments as well.*
>
> *(p 16) A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed. ...*

Constructive Mathematics and Computer Programming

(p 21-22) Now to the rules of inference or proof rules, as they are called in programming.

...

For each of these rules of inference, the reader is asked to try to make the conclusion evident to himself on the presupposition that he knows the premises. This does not mean that further verbal explanations are of no help in bringing about an understanding of the rules, only that this is not the place for such detailed explanations. But there are also certain limits to what verbal explanations can do when it comes to justifying axioms and rules of inference. In the end, everybody must understand for himself.

Meanings and interpretations

The crucial word is "*means*" as opposed to

- "is interpretated as"
- "is modelled as"

in the sense of *metamathematics*: "a rigorous mathematical technique for investigating a great variety of foundation problems for mathematics and logic" (Kleene 1952, p. 59). Formal metamathematical reasoning takes place in a *metalanguage*, which needs to be stronger than the object language.

What are meaning explanations?

- Synonyms: *direct, intuitive*, or *standard* semantics of intuitionistic type theory. Also *syntactico-semantical approach* to meaning.
- This is *pre-mathematical* semantics is to be contrasted to meta-mathematical semantics.

Meaning explanations: pre-mathematical or metamathematical?

Martin-Löf: *Intuitionistic Type Theory*, Bibliopolis, 1984, p 1, par 1.

> *"Mathematical logic and the relation between logic and mathematics have been interpreted in at least three different ways:*
>
> 1. *mathematical logic as symbolic logic, or logic using mathematical symbolism;*
> 2. *mathematical logic as foundations (or philosophy) of mathematics;*
> 3. *mathematical logic as logic studied by mathematical methods, as a branch of mathematics.*
>
> *We shall here mainly be interested in mathematical logic in the second sense. What we shall do is also mathematical logic in the first sense, but certainly not in the third."*

Milestones in the foundations of intuitionistic logic

BHK: Brouwer 1908, Heyting, Kolmogorov 1925: A calculus of problems - pre-mathematical "meaning explanations" for intuitionistic predicate logic.

Kleene 1945: Recursive realizability. A number e realizes a formula ϕ – metamathematical interpretation of intuitionistic arithmetic.

Curry-Howard-de Bruijn 1969: Propositions as types. $a : A$ can be read as the term a is a proof of the proposition A. Martin-Löf 1972: Intuitionistic type theory. A formal system.

Martin-Löf 1979: Pre-mathematical meaning explanations for intuitionistic type theory.

The meaning of implication

Kolmogorov: $\phi \supset \psi$ means that a solution of the problem ψ can be reduced to a solution of the problem ϕ.

Kleene: The number e realizes $\phi \supset \psi$ iff for all e' realizing ϕ we have $\{e\}e'$ realizing ψ. Here e is a numerical code for a Turing machine, and $\{e\}e'$ is the result of running that machine with input e'.

Curry-Howard: The typing $f : A \to B$ is derivable, and represents a proof of the proposition $\phi \supset \psi$ (a solution of the problem).

Martin-Löf: $f : A \to B$ means that f has canonical form $(\lambda x)b$ and for all $a : A$ we have $b[a/x] : B$.

Intuitionistic Type Theory

A (series of) formal logical theory(ies) introduced by Per Martin-Löf:

1972 (1997)	*An intuitionistic theory of types*
1973 (1975)	*An intuitionistic theory of types: predicative part*

Present theories with normalization proofs and corollaries like decidability of the judgment $a : A$. Intensional type theory!

1979 (1982)	*Constructive mathematics and computer programming*
1980 (1984)	*Intuitionistic Type Theory*, book published by Bibliopolis

Present theories with "meaning explanations". Justify equality reflection and uniqueness of identity types. Extensional type theory! Normalization and decidability fail.

1986 (1989)	The theory based on a logical framework. An intensional type theory! The basis of Agda (roughly).

Why are the meaning explanations important?

- Constructive mathematics = computer programming. Identity of mathematical correctness and program correctness. More rigorous and more general than BHK.
- Foundations for CZF and foundational systems based on untyped lambda-calculus and cominatory logic (Feferman's explicit mathematics, Aczel's logical theory of constructions).
- Guidelines when extending intuitionistic type theory.
  - Extensions with proof-theoretically stronger principles: superuniverse, Mahlo-universe, induction-recursion, autonomous Mahlo-universe, etc.
  - Computational content of classical logic
  - Homotopy type theory!
  - Proof assistants (Agda, etc).
- Impredicative systems?? Calculus of (Inductive) Constructions (Coq)

Meaning according to "Constructive Mathematics and Computer Programming" (1979)

Technically, a variant of realizability interpretation, where realizers are lambda terms:

- Terms (realizers); essentially untyped lambda terms with constants, including terms denoting types.
- Canonical terms: closed terms beginning with a constructor:

$$(\lambda x)b, (a,b), \mathrm{inl}(a), \mathrm{inr}(b), \mathrm{refl}, 0, \mathrm{suc}(a), \ldots,$$

$$(\Pi x : A)B, (\Sigma x : A)B, A+B, \mathrm{Id}(A,a,b), \mathsf{N}, \mathsf{U}, \ldots$$

  (whnfs, lazy values).
- Computation relation $a \Rightarrow v$ between a *closed* term a and its canonical form v.

Natural numbers - computation to canonical form

Start with *untyped computation system* with notion of computation of *closed* expression to *canonical form* (whnf) $a \Rightarrow v$.

$$\mathsf{N} \Rightarrow \mathsf{N}$$

$$0 \Rightarrow 0 \qquad\qquad \mathrm{suc}(a) \Rightarrow \mathrm{suc}(a)$$

$$\frac{c \Rightarrow 0 \quad d \Rightarrow v}{\mathcal{R}\,c\,d\,e \Rightarrow v} \qquad\qquad \frac{c \Rightarrow \mathrm{suc}\,a \quad e\,a\,(\mathcal{R}\,a\,d\,e) \Rightarrow v}{\mathcal{R}\,c\,d\,e \Rightarrow v}$$

Natural numbers - meaning explanations

$$\frac{A \Rightarrow \mathsf{N}}{A\ type}$$

$$\frac{A \Rightarrow \mathsf{N} \quad A' \Rightarrow \mathsf{N}}{A = A'}$$

$$\frac{A \Rightarrow \mathsf{N} \quad a \Rightarrow 0}{a : A} \qquad \frac{A \Rightarrow \mathsf{N} \quad a \Rightarrow \mathrm{suc}(b) \quad b : \mathsf{N}}{a : A}$$

$$\frac{A \Rightarrow \mathsf{N} \quad a \Rightarrow 0 \quad a' \Rightarrow 0}{a = a' : A} \qquad \frac{A \Rightarrow \mathsf{N} \quad a \Rightarrow \mathrm{suc}(b) \quad a' \Rightarrow \mathrm{suc}(b') \quad b = b' :}{a = a' : A}$$

How to understand these rules, metamathematically (realizability) or pre-mathematically (meaning explanations)?

Function types - meaning explanations

$$\frac{A \Rightarrow (\Pi y : B)\, C \quad B \; type \quad y : B \vdash C \; type}{A \; type}$$

$$\frac{A \Rightarrow (\Pi y : B)\, C \quad A' \Rightarrow (\Pi y : B')\, C' \quad B = B' \quad y : B \vdash C = C'}{A = A'}$$

$$\frac{A \Rightarrow (\Pi y : B)\, C \quad a \Rightarrow (\lambda y)\, c \quad y : B \vdash c : C}{a : A}$$

$$\frac{A \Rightarrow (\Pi y : B)\, C \quad a \Rightarrow (\lambda y)\, c \quad a' \Rightarrow (\lambda y)\, c' \quad y : B \vdash c = c' : C}{a = a' : A}$$

Note: reference to the *hypothetical* judgments

$$y : B \vdash C \; type \quad y : B \vdash C = C'$$

$$y : B \vdash c : C \quad y : B \vdash c = c' : C$$

Hypothetical judgments - meaning explanations

Martin-Löf 1982:

- $y : B \vdash C\ type$ means that if $b : B$ then $C[b/y]\ type$
- $y : B \vdash C = C'$ means that if $b : B$ then $C[b/y] = C'[b/y]$
- $y : B \vdash c : C$ means that if $b : B$ then $c[b/y] : C[b/y]$
- $y : B \vdash c = c' : C$ means that if $b : B$ then $c[b/y] = c'[b/y] : C[b/y]$

If we interpret the meaning explanation rules as the introduction rules for a mutual inductive definition of the different judgment forms, we get a *negative* inductive definition!

It's an *inductive-recursive* definition.

But an inductive-recursive definition is a (meta)mathematical notion, not a premathematical one.

What is a mathematical object? The general pattern.

Let C be an n-ary type constructor. We have the following semantic rule:

$$\frac{A \Rightarrow C(a_1, \ldots, a_n) \quad \cdots}{A \ type}$$

Let c be an m-ary term constructor for C. We have the following semantic rule

$$\frac{A \Rightarrow C(a_1, \ldots, a_n) \quad a \Rightarrow c(b_1, \ldots, b_m) \quad \cdots}{a : A}$$

The dots $(\cdots)$ specify the types of $a_1, \ldots, a_n, b_1, \ldots, b_m$.

Identity types

$$\frac{A \Rightarrow \mathrm{Id}(B,b,b') \quad B\ \mathit{type} \quad b,b' : B}{A\ \mathit{type}}$$

$$\frac{A \Rightarrow \mathrm{Id}(B,b,b') \quad a \Rightarrow \mathsf{refl} \quad B\ \mathit{type} \quad b,b' : B \quad b = b' : B}{a : A}$$

Agda code

`SyntaxComputationRules.agda` Terms and canonical terms (values). Substitution. Computation rules.

`MeaningOfJudgments.agda` Inductive-recursive definition of the canonical types and their canonical terms. We here use that something is a type if it has a canonical type as value and that something is a term of a certain type if it has a canonical term of the canonical type in question as value.

`JustificationOfRules.agda` Justification of some rules for natural numbers.

The premathematical/direct reading

- A grammar: a prescription for how to generate strings in a language (the mathematical reading is an inductive definition of a set of strings).
- Computation rules: a prescription for how to compute the canonical form of a program (the mathematical reading is an inductive definition of a set of strings)
- The meaning of the typing judgment: a prescription for how to compute canonical forms and check whether a term constructor matches a type constructor, and then continue with the subgoals (the mathematical reading is as an inductive-recursive definition of two relations).
- Justification of the rules: carrying out type-checking experiments, or thinking about the meaning (mathematical reading gives proofs in the metalanguage)