# Program Testing and Constructive Validity

Peter Dybjer

Chalmers University of Technology, Göteborg, Sweden

Philosophy and Foundations of Mathematics:
Epistemological and Ontological Aspects
-
to Per Martin-Löf on the occasion of his retirement

Swedish Collegium for Advanced Study, Uppsala, May 5-8, 2009

## A comment on program proving

*When you've proved your program correct, then you'd better run it, to make sure it works!*

## A comment on program proving

*When you've proved your program correct, then you'd better run it, to make sure it works!*

How come?

## A comment on program proving

*When you've proved your program correct, then you'd better run it, to make sure it works!*

How come?

- wrong specification?
- wrong model of program?
- wrong manual proof?
- mechanical proof, but faulty proof assistant?

## A comment on program proving

*When you've proved your program correct, then you'd better run it, to make sure it works!*

How come?

- wrong specification?
- wrong model of program?
- wrong manual proof?
- mechanical proof, but faulty proof assistant?

What does this have to do with the foundations of mathematics?

## A comment on the validity of the logical laws

*When you've made your logical law evident to yourself, then you'd better run it, to make sure it's valid!*

How can you "run" a logical formula?

## A comment on the validity of the logical laws

*When you've made your judgement evident to yourself,*
*then you'd better run it, to make sure it's valid!*

This is possible in Martin-Löf type theory, in a sense to be explained.

## A comment on the validity of the logical laws

*When you've made your judgement evident to yourself,*
*then you'd better run it, to make sure it's valid!*

This is possible in Martin-Löf type theory, in a sense to be explained.

The meaning explanations in "Constructive Mathematics and Computer Programming" (1979) from the point of view of the computer programmer (or perhaps better, the computer "user"), rather than from the point of view of the constructive mathematician.

## Meaning explanations for intuitionistic type theory

Meaning explanations for *extensional polymorphic type theory*.

References by Martin-Löf:

Hannover 1979 (1982) *Constructive mathematics and computer programming*

Padova 1980 (1984) *Intuitionistic Type Theory*, book published by Bibliopolis

## What are Martin-Löf’s meaning explanations?

Meaning explanations. Also called

> *direct semantics, intuitive semantics, standard semantics, syntactico-semantical approach*

They are "pre-mathematical" as opposed to "meta-mathematical":

> *mathematical semantics (assuming set theory as meta-language)*

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

Before 1979: normalization proofs, but no meaning explanations.

## Meta-mathematical interpretation of meaning explanations

"It's just realizability!"

## Meta-mathematical interpretation of meaning explanations

"It's just realizability!"

A special kind of abstract realizability:

- Realizers are lambda terms:

  $$a ::= x \mid (x)a \mid a(a) \mid \lambda \mid Ap \mid 0 \mid s \mid R \mid r \mid J \mid \Pi \mid N \mid I \mid U \mid \cdots$$

  Some terms denote types.
- Judgements are interpreted in terms of the relation $a \Rightarrow v$ between *closed* terms, meaning "$a$ has canonical form $v$". Canonical forms are

  $$v ::= \lambda(a) \mid 0 \mid s(a) \mid r \mid \Pi(a,a) \mid N \mid I(a,a,a) \mid U \mid \cdots$$

(We write $f(a_1,\ldots,.a_n) = f(a_1)\cdots(a_n)$ and $(\lambda x)a = \lambda((x)a)$.)

## General pattern

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

$$\frac{A \Rightarrow C(a_1, \ldots . a_m) \quad A' \Rightarrow C(a'_1, \ldots . a'_m) \quad \cdots}{A = A'}$$

where $C$ is an $m$-place type constructor, and

$$\frac{A \Rightarrow C(a_1, \ldots . a_m) \quad a \Rightarrow c(b_1, \ldots . b_n) \quad \cdots}{a \in A}$$

$$\frac{A \Rightarrow C(a_1, \ldots . a_m) \quad a \Rightarrow c(b_1, \ldots . b_n) \quad a' \Rightarrow c(b'_1, \ldots . b'_n) \quad \cdots}{a = a' \in A}$$

where $c$ is an $n$-place term constructor for $C$.

## Natural numbers

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

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

$$\frac{A \Rightarrow N \quad a \Rightarrow 0}{a \in A} \qquad \frac{A \Rightarrow N \quad a \Rightarrow s(b) \quad b \in N}{a \in A}$$

$$\frac{A \Rightarrow N \quad a \Rightarrow 0 \quad a' \Rightarrow 0}{a = a' \in A} \qquad \frac{A \Rightarrow N \quad a \Rightarrow s(b) \quad a' \Rightarrow s(b') \quad b = b' \in N}{a = a' \in A}$$

## Meta-mathematical interpretation of meaning explanations

A partial equivalence relation (per) model!

We simultaneously define the following relations on the set of closed terms:

- the per of "equal types" $A = A'$
- the family of pers of "equal terms of a given type" $a = a' \in A$.

The rules in previous slides inductively generate these pers, Allen (1987). Related models can be found in Aczel (1974, 1980), Beeson (1982), Smith (1984).

## Meta-mathematical interpretation of meaning explanations

A partial equivalence relation (per) model!

We simultaneously define the following relations on the set of closed terms:

- the per of "equal types" $A = A'$
- the family of pers of "equal terms of a given type" $a = a' \in A$.

The rules in previous slides inductively generate these pers, Allen (1987). Related models can be found in Aczel (1974, 1980), Beeson (1982), Smith (1984).

However, the idea that meaning explanations are "just realizability" is, although helpful, fundamentally misleading!

## The meaning of induction

How is now the rule of N-elimination justified?

$$\frac{C(x)\ \mathit{type} \qquad n \in N \qquad d \in C(0) \qquad e \in (\Pi x \in N)C(x) \to C(s(x))}{R(n, d, e) \in C(n)}$$

## The meaning of induction

How is now the rule of N-elimination justified?

$$\frac{C(x)\ \mathit{type} \qquad n \in N \qquad d \in C(0) \qquad e \in (\Pi x \in N)C(x) \to C(s(x))}{R(n, d, e) \in C(n)}$$

By mathematical induction on the meta-level (in set theory)!

## The meaning of induction

How is now the rule of N-elimination justified?

$$\frac{C(x)\ type \qquad n \in N \qquad d \in C(0) \qquad e \in (\Pi x \in N)C(x) \to C(s(x))}{R(n, d, e) \in C(n)}$$

By mathematical induction on the meta-level (in set theory)!

This is not a satisfactory explanation for a constructivist! Let us work with an intuitionistic metalanguage. The model construction refered to above can be carried out in such a language (Aczel 1983, Smith 1984).

## The meaning of induction

How is now the rule of N-elimination justified?

$$\frac{C(x)\ \mathit{type} \qquad n \in N \qquad d \in C(0) \qquad e \in (\Pi x \in N) C(x) \to C(s(x))}{R(n, d, e) \in C(n)}$$

By mathematical induction on the meta-level (in set theory)!

This is not a satisfactory explanation for a constructivist! Let us work with an intuitionistic metalanguage. The model construction refered to above can be carried out in such a language (Aczel 1983, Smith 1984).

Why is the rule of induction on the intuitionistic meta-level correct?

## The meaning of induction

How is now the rule of N-elimination justified?

$$\frac{C(x)\ \mathit{type} \qquad n \in N \qquad d \in C(0) \qquad e \in (\Pi x \in N) C(x) \to C(s(x))}{R(n, d, e) \in C(n)}$$

By mathematical induction on the meta-level (in set theory)!

This is not a satisfactory explanation for a constructivist! Let us work with an intuitionistic metalanguage. The model construction refered to above can be carried out in such a language (Aczel 1983, Smith 1984).

Why is the rule of induction on the intuitionistic meta-level correct?

Because of the BHK-interpretation, i e, the rule of N-elimination. Etc.

## The correctness of powerful induction principles

How do we justify the rules for Setzer's $\Pi_3$-reflecting universe?

## The correctness of powerful induction principles

How do we justify the rules for Setzer’s $\Pi_3$-reflecting universe?

By the intuitionistically valid $\Pi_3$-reflecting universe principle on the meta-level!

## Mathematical induction and inductive inference

Alternatively. The rule of induction is correct because if we test the primitive recursion combinator $R(n,d,e)$ for $n = 0,1,2,\ldots,$ and for arbitrary base case $d$ and arbitrary step case $e$, it succeeds each time (more later)!

## Mathematical induction and inductive inference

Alternatively. The rule of induction is correct because if we test the primitive recursion combinator $R(n, d, e)$ for $n = 0, 1, 2, \ldots,$ and for arbitrary base case $d$ and arbitrary step case $e$, it succeeds each time (more later)!

Inductive inference!

## Mathematical induction and inductive inference

Alternatively. The rule of induction is correct because if we test the primitive recursion combinator $R(n, d, e)$ for $n = 0, 1, 2, \ldots,$ and for arbitrary base case $d$ and arbitrary step case $e$, it succeeds each time (more later)!

Inductive inference!

Similarly, the elimination rules for the $\Pi_3$-reflecting universe are correct because if we test them for suitable inputs these tests succeed!

## Static vs dynamic, time vs space

## Static vs dynamic, time vs space

The meaning explanations are about what *really* happens!

## Static vs dynamic, time vs space

The meaning explanations are about what *really* happens!

Syntactico-semantical approach! Semantics is what happens during execution. Meaning = extension.

## Static vs dynamic, time vs space

The meaning explanations are about what *really* happens!

Syntactico-semantical approach! Semantics is what happens during execution. Meaning = extension.

$a \Rightarrow b$ is a static mathematical representation of the *real* fact that $a$ will turn into $b$ a little later after some computation is done.

## Pre-mathematical rendering of meaning explanations

Read the rules (which are said to "inductively generate" the realizability interpretation), rather as a "testing manual", a manual for "falsification of conjectures", or "bug-finding". A tester only needs to be able to push a button "execute program" and inspect results. He/she is only a "user" who does not need to know logic or programming.

How do you test that

$$A\ \mathit{type}?$$

$$A = A?'$$

$$a \in A?$$

$$a = a' \in A?$$

## How to test *A type*?

Some rules for deriving judgements of the form *A type* are

$$\frac{A \Rightarrow N}{A \ type} \qquad \frac{A \Rightarrow I(B, b, b') \quad B \ type \quad b \in B \quad b' \in B}{A \ type} \qquad \frac{A \Rightarrow U}{A \ type}$$

The testing manual reading: Run $A$!

- If it has canonical form $N$, then the test is successful.
- If it has canonical form $I(B, b, b')$, then first test $B$ *type* and if successfull test $b \in B$ and then $b' \in B$.
- If it has canonical form $U$, then the test is successful.
- If it has a canonical form which does not begin with a type constructor, or no canonical form at all, then the test fails.

## How to test $a \in A$?

$$\frac{A \Rightarrow N \quad a \Rightarrow 0}{a \in A} \qquad \frac{A \Rightarrow N \quad a \Rightarrow s(b) \quad b \in N}{a \in A}$$

$$\frac{A \Rightarrow I(B, b, b') \quad a \Rightarrow r \quad B \textit{ type} \quad b, b' \in B \quad b = b' \in B}{a \in A}$$

$$\frac{A \Rightarrow U \quad a \Rightarrow N}{a \in A} \qquad \frac{A \Rightarrow U \quad a \Rightarrow I(B, b, b') \quad B \in U \quad b, b' \in B}{a \in A}$$

Testing manual: Run $A$ and $a$! Depending on their canonical form continue with the tests prescribed by the remaining premises of the appropriate rule!

## Testing hypothetical judgements

How do we read

$$\frac{A \Rightarrow \Pi(B, C) \qquad a \Rightarrow \lambda(c) \qquad x \in B \vdash c(x) \in C(x)}{a \in A}$$

as a rule in our testing manual? What action should we take to test

$$x \in B \vdash c(x) \in C(x)?$$

## Testing hypothetical judgements

How do we read

$$\frac{A \Rightarrow \Pi(B,C) \qquad a \Rightarrow \lambda(c) \qquad x \in B \vdash c(x) \in C(x)}{a \in A}$$

as a rule in our testing manual? What action should we take to test

$$x \in B \vdash c(x) \in C(x)?$$

Maybe: for all $b \in B$ it is the case that $c(b) \in C(b)$?

## Testing hypothetical judgements

How do we read

$$\frac{A \Rightarrow \Pi(B, C) \qquad a \Rightarrow \lambda(c) \qquad x \in B \vdash c(x) \in C(x)}{a \in A}$$

as a rule in our testing manual? What action should we take to test

$$x \in B \vdash c(x) \in C(x)?$$

Maybe: for all $b \in B$ it is the case that $c(b) \in C(b)$? But how did we get $b \in B$?

## Testing hypothetical judgements

How do we read

$$\frac{A \Rightarrow \Pi(B, C) \qquad a \Rightarrow \lambda(c) \qquad x \in B \vdash c(x) \in C(x)}{a \in A}$$

as a rule in our testing manual? What action should we take to test

$$x \in B \vdash c(x) \in C(x)?$$

Maybe: for all $b \in B$ it is the case that $c(b) \in C(b)$? But how did we get $b \in B$? Maybe from a maliscious hacker?

## Testing hypothetical judgements

How do we read

$$\frac{A \Rightarrow \Pi(B, C) \qquad a \Rightarrow \lambda(c) \qquad x \in B \vdash c(x) \in C(x)}{a \in A}$$

as a rule in our testing manual? What action should we take to test

$$x \in B \vdash c(x) \in C(x)?$$

Maybe: for all $b \in B$ it is the case that $c(b) \in C(b)$? But how did we get $b \in B$? Maybe from a maliscious hacker?

We'd better manufacture our own tests! The rules which are said to "inductively generate" the realizability interpretation is now given a second reading: how to generate input to hypothetical tests!
This is a key point!

## How to generate $x \in A$?

Read the rules as a manual (for the user) for generating an input $x$ :

$$\frac{A \Rightarrow N \quad x \Rightarrow 0}{x \in A} \qquad \frac{A \Rightarrow N \quad x \Rightarrow s(y) \quad y \in N}{x \in A}$$

$$\frac{A \Rightarrow I(B, b, b') \quad B \textit{ type} \quad b \in B \quad b' \in B \quad b = b' \in B \quad x \Rightarrow r}{x \in A}$$

$$\frac{A \Rightarrow U \quad x \Rightarrow N}{x \in A} \qquad \frac{A \Rightarrow U \quad x \Rightarrow I(Y, y, y') \quad Y \in U \quad y \in Y \quad y' \in Y}{x \in A}$$

Input generation manual: Run $A$!

- If it has canonical form $N$, then either generate $x = 0$ or generate $x = s(y)$ and then, if necessary, generate $y \in N$.
- If it has canonical form $I(B, b, b')$? To be discussed later.
- If it has canonical form $U$, then either generate $x = N$ or generate $x = I(Y, y, y')$ and then, if necessary, generate $Y \in U, y \in Y, y' \in Y$.

## Functional input. How to generate $x \in N \longrightarrow N$?

How do we read the rule

$$\frac{x \Rightarrow \lambda(z) \quad y \in N \vdash z(y) \in N}{x \in N \longrightarrow N}$$

as a rule for generating $x$?
It would be wrong to try to read $y \in N \vdash z(y) \in N$ as syntactic derivability in some formal system for Martin-Löf type theory. We want the *semantic* notion! Cf discussion of alleged impredicativity of functionals. We want it to be "local", that is, the test generated does not depend on the formal system as a whole.

## Domain theory to the rescue

Domain theory (continuity principle) and game semantics to the rescue! We generate input/output pairs $(m, n)$ with $m, n \in N$. Which ones? As many as needed! But we do not know in advance. When we test e g

$$x \in N \rightarrow N \vdash b(x) \in B(x)$$

We will have to begin to test $b(x) \in B(x)$ without knowing $x$. At some stage the computation gets stuck because it does not know $x$. E g $R(Ap(x, 0), d, e)$ needs to know the canonical form of $Ap(x, 0)$. We generate an input output pair $(0, y)$ for $x$ where $y \in N$ is generated as before. Now the computation can go on. Etc.

## Identity types. How to generate $x \in I(N, a, b)$ and $x \in I(N \longrightarrow N, f, g)$?

How do we read the rule

$$\frac{A \Rightarrow I(N, m, n) \quad N\ type \quad m, n \in N \quad m = n \in N \quad x \Rightarrow r}{x \in A}$$

as a rule for generating $x$? If $m = n \in N$ (closed expressions), then generate $x = r$, otherwise, there is no x!
How do we read the rule

$$\frac{A \Rightarrow I(N \longrightarrow N, f, g) \quad N \longrightarrow N\ type \quad f, g \in N \longrightarrow N \quad f = g \in N \longrightarrow N \quad x \Rightarrow r}{x \in A}$$

as a rule for generating $x$? If $f = g \in N \longrightarrow N$ (closed expressions), then generate $x = r$, otherwise, there is no x! But we will not be able to establish this in finite time! What to do?

## Meaning explanations for impredicative type theory?

What about testing manuals for

- System F (Girard)?
- Calculus of Constructions (Coquand and Huet)?
- Calculus of Inductive Constructions (Coquand and Paulin), the theory of the Coq-system?

## Meaning explanations for impredicative type theory?

What about testing manuals for

- System F (Girard)?
- Calculus of Constructions (Coquand and Huet)?
- Calculus of Inductive Constructions (Coquand and Paulin), the theory of the Coq-system?

These systems are justified with normalization proofs, in the style of Martin-Löf type theory prior to 1979.

## Meaning explanations for impredicative type theory?

What about testing manuals for

- System F (Girard)?
- Calculus of Constructions (Coquand and Huet)?
- Calculus of Inductive Constructions (Coquand and Paulin), the theory of the Coq-system?

These systems are justified with normalization proofs, in the style of Martin-Löf type theory prior to 1979. Will they see a 1979?

## Meaning explanations for impredicative type theory?

What about testing manuals for

- System F (Girard)?
- Calculus of Constructions (Coquand and Huet)?
- Calculus of Inductive Constructions (Coquand and Paulin), the theory of the Coq-system?

These systems are justified with normalization proofs, in the style of Martin-Löf type theory prior to 1979. Will they see a 1979?
These systems, especially the last one, have real users. What do they expect when they "run" their programs?

## An impredicative universe

Predicative universe closed under Π:

$$\frac{A \in U \quad x \in A \vdash B \in U}{(\Pi x \in A)B \in U}$$

Impredicative universe *U* closed under Π:

$$\frac{A \textit{ type} \quad x \in A \vdash B \in U}{(\Pi x \in A)B \in U}$$

e g

$$N = (\Pi X \in U)X \to (X \to X) \to X \in U$$

$$I(A, a, b) = (\Pi X \in A \to U)X(a) \to X(b) \in U$$

## Testing manual for the Calculus of Constructions

Formally, there is great similarity between Martin-Löf type theory and the Calculus of Constructions, except that the latter

- only has types $U$ and $(\Pi x \in B)C$, no primitive data types $N, I(A, a, b), \ldots$
- $U$ is closed under impredicative $\Pi$

Modify the testing manual for Martin-Löf type theory accordingly. The difference appears in the test for elements of $U$. The only rule is:

$$\frac{A \Rightarrow U \quad a \Rightarrow (\Pi x \in B)C \quad B\ type \quad x \in B \vdash C \in U}{a \in A}$$

No base case!

## Testing based on normalization of open expressions?

Per Martin-Löf (2009): *Evaluation of open expressions*

> *The informal, or intuitive, semantics of type theory makes it evident that closed expressions of ground type evaluate to head normal form, whereas metamathematics, ..., is currently needed to show that expressions which are open or of higher type can be reduced to normal form. The question to be discussed is: Would it be possible to modify the informal semantics in such a way that it becomes evident that all expressions, also those that are open or of higher type, can be reduced to full normal form?*

The user computes with open expressions in CoC. Open weak head normal forms are

$$v ::= U \,|\, (\Pi x \in a)a \,|\, (\lambda x)a \,|\, x(a, \ldots, a)$$

What about CIC?

## Testing manual for CoC

Testing types

$$\frac{A \Rightarrow (\Pi y \in B)C \quad \Gamma \vdash B \; type \quad \Gamma, y \in B \vdash C \; type}{\Gamma \vdash A \; type} \qquad \frac{A \Rightarrow U}{\Gamma \vdash A \; type}$$

Testing terms

$$\frac{A \Rightarrow U \qquad a \Rightarrow (\Pi y \in B)C \qquad \Gamma \vdash B \; type \qquad \Gamma, y \in B \vdash C \in U}{\Gamma \vdash a \in A}$$

$$\frac{A \Rightarrow (\Pi x \in B)C \qquad a \Rightarrow (\lambda y)c \qquad \Gamma, y \in B \vdash c \in C}{\Gamma \vdash a \in A}$$

## Testing manual for CoC; neutral forms

$$\frac{a \Rightarrow x(b_1, \ldots, b_n) \qquad x \in (\Pi y_1 : B_1) \cdots (\Pi y_n : B_n) C \text{ in } \Gamma}{\dfrac{\Gamma \vdash b_1 \in B_1 \quad \cdots \quad \Gamma \vdash b_n \in B_n \quad \Gamma \vdash C[b_1/y_1, \ldots, b_n/y_n] = A}{\Gamma \vdash a \in A}}$$

Testing manual = type checking algorithm ...

## Summary

Some important distinctions:

- pre-mathematical vs meta-mathematical
- local (single judgement) vs
  global (system of inference rules) correctness
- judgement vs proposition
- validated by testing (objective) vs
  made evident by thinking (subjective)
- game semantics vs realizability semantics
- input generation vs output computation
- primary school computation of closed expressions vs
  secondary school computation of open expressions

## Logical rule vs its implementation

Q: You test the implementation of a judgement, not the judgement itself
A: It is only meaningful to test an implementation, if by that we mean something which so as to speak can run by itself, something mechanical, a machine, or a user without knowledge of logic.

## Incompleteness of testing

Q: In this way you will never be able to establish validity of a judgement!
A: I'm not saying that the only way we can establish validity of a judgement is to test it. I'm only saying that testing is the ultimate verification whether you were correct when you said that a certain judgement is evident. That's why the first slide is the most important one.

## Testing the Ackermann function

Q: It doesn't make sense to say that you can test the Ackermann function. Very quickly the computations will take too long for you to see the result, although they will in fact terminate.
A: Although it is not possible to observe non-termination, nothing forbids us to look at intermediate stages of a computation and realize that it has got into a loop, or an infinite regress that it will never get out of. This is a good falsification.

## Testing formulas in classical set theory

Q: I can test a formula in set theory too! I just enumerate all the theorems, and test whether a particular one is in the enumeration!
A: Yes, but this is "global" testing, testing of the whole system. Testing of judgements in type theory are "local", you only test the judgement itself, not the whole system of which it is a rule.

## What does "A type" mean?

Q: To know that A is a type, you have to know how the elements are formed and how equal elements are formed. You only say "A has to have canonical type as value". Why?
A: It's part of the assumptions we have before we do the testing that we know the introduction rules for each type former.