The Evolution of Inductive Definitions in Type Theory (a Retrospective)

to Christine
on the occasion of her honorary doctorate
at Gothenburg University

Peter Dybjer

Workshop on Proofs and Programs
22 October, 2011

Some papers on inductive definitions by Christine

- "Extraction de Programmes dans le Calcul des Constructions" (PhD thesis 1989)
- "Inductively Defined Types in the Calculus of Constructions" (MFPS 1989) with Frank Pfenning
- "Inductive Types" (COLOG-88) with Thierry Coquand
- "Inductive Definitions in the system Coq - Rules and Properties" (TLCA 1993)

Intuitionistic type theory - before 1984

1971 Intuitionistic type theory with *type* : *type* - impredicative and inconsistent

1972 Intuitionistic type theory - predicative, intensional and consistent

1979 Intuitionistic type theory - predicative, extensional and with meaning explanations "Constructive Mathematics and Computer Programming". Application to computer science started shortly afterwards in Gothenburg and at Cornell.

1982-83 ca First proof assistants for intuitionistic type theory (GTT and NuPRL)

What is a mathematical object, constructively?

Martin-Löf type theory is not just a full-scale logical system for constructive mathematics – it comes with a "meaning theory".

What is a mathematical object, constructively?

Martin-Löf type theory is not just a full-scale logical system for constructive mathematics – it comes with a "meaning theory".

- What kind of thing is a computable function?

What is a mathematical object, constructively?

Martin-Löf type theory is not just a full-scale logical system for constructive mathematics – it comes with a "meaning theory".

- What kind of thing is a computable function?
- What kind of things are the inputs and outputs of computable functions? Numbers? Unary or binary?

What is a mathematical object, constructively?

Martin-Löf type theory is not just a full-scale logical system for constructive mathematics – it comes with a "meaning theory".

- What kind of thing is a computable function?
- What kind of things are the inputs and outputs of computable functions? Numbers? Unary or binary?
  Instead inputs and outputs of computable functions are structured objects: numbers, functions, pairs, lists, trees, ...

Types of mathematical objects in intuitionistic type theory

1972 (97): $(\Pi x : A)B(x), (\Sigma x : A)B(x), A + B, \mathrm{N}, \mathrm{N}_n, \mathrm{U}$

1973 (75): add $\mathrm{I}(A, a, b), \mathrm{U}_n$

1979 (82): add $(\mathrm{W}x : A)B(x)$

1980 (84): add $\mathcal{O}, \mathrm{List}(A)$, (universes a la Tarski)

An open system

New types can be added whenever there is a need for them, provided meaning explanations can be provided for them, see for example, Nordström "Multilevel Functions in Martin-Löf's Type Theory" 1985.
The general principle is that mathematical objects are "inductively generated". But what does this mean?

An open system

New types can be added whenever there is a need for them, provided meaning explanations can be provided for them, see for example, Nordström "Multilevel Functions in Martin-Löf's Type Theory" 1985.
The general principle is that mathematical objects are "inductively generated". But what does this mean?
Or that mathematical objects are "well-founded trees". But what does this mean?

An open system

New types can be added whenever there is a need for them, provided meaning explanations can be provided for them, see for example, Nordström "Multilevel Functions in Martin-Löf's Type Theory" 1985.
The general principle is that mathematical objects are "inductively generated". But what does this mean?
Or that mathematical objects are "well-founded trees". But what does this mean?
Can one say something more precise about when it is correct to add a new type of objects to intuitionistic type theory?

An open system

New types can be added whenever there is a need for them, provided meaning explanations can be provided for them, see for example, Nordström "Multilevel Functions in Martin-Löf's Type Theory" 1985.
The general principle is that mathematical objects are "inductively generated". But what does this mean?
Or that mathematical objects are "well-founded trees". But what does this mean?
Can one say something more precise about when it is correct to add a new type of objects to intuitionistic type theory?
Then there was 1984.

The Calculus of Constructions (1984)

CC has impredicative universe $*$ closed under dependent function space:

$$\frac{A \ \mathit{type} \quad x : A \vdash B : *}{(x : A) \to B : *}$$

Types of Church encodings

$$N = (X : *) \to X \to (X \to X) \to X : *$$

$$I\, A\, a\, b = (X : A \to *) \to X\, a \to X\, b : *$$

Cf predicative universe of Martin-Löf type theory closed under dependent function space:

$$\frac{A : * \quad x : A \vdash B : *}{(x : A) \to B : *}$$

CC can encode inductive families

In a joint paper with Frank Pfenning (MFPS 1989) Christine formulated the following type constructor

$$\begin{array}{lll} \textbf{indtype}\ \alpha & : & (z_1 : Q_1) \to \cdots \to (z_m : Q_m) \to * \ \textbf{with} \\ & & \vdots \\ & & c : (x_1 : P_1) \to \cdots \to (x_k : P_k) \to \alpha\, M_1 \cdots M_m \\ & & \vdots \\ & & \textbf{end} \end{array}$$

Restrictions:

- α may not occur in Q_i.
- α may only occur *positively* in P_j.

CC can encode inductive families

Associate with each inductively defined type α a type $\underline{\alpha}$ in the pure CC by a systematic impredicative encoding.

Theorem (Adequacy of impredicative encodings): Bijection between equivalence classes of terms in $\alpha\, M_1 \cdots M_m$ and $\underline{\alpha}\, M_1 \cdots M_m$

In CC all mathematical objects are (coded as) lambda terms (Church numerals, Church truth values, etc)!

A problem: nonderivability of Induction in CC

If

$$n : N = (X : *) \to X \to (X \to X) \to X$$

$$Ind\, n = (\mathrm{C} : N \to *) \to C\, 0 \to ((x : N) \to C\, x \to C\, (succ\, x)) \to C\, n$$

then the induction principle

$$(n : N) \to Ind\, n$$

is not derivable in CC.

Note that

$$(X : *) \to ((X \to X) \to X) \to X : *$$

is a well-formed type in CC. What is the induction principle?

Non-derivability of induction for arbitrary encoding

Geuvers (TLCA 2001): In CC there is no instantiation of the context

$$N : *,\ 0 : N,\ s : N \to N,\ R : (n : N) \to Ind\ n$$

Assuming the induction principle

Instead one assumes the induction principle for N.

Assuming the induction principle

Instead one assumes the induction principle for N.

New problem: how to prove $0 \neq 1$?

Assuming the induction principle

Instead one assumes the induction principle for N.

New problem: how to prove $0 \neq 1$? Extend CC with universes.

Extending CC with primitive inductive types - CIC

Extend CC with rules for primitive inductive types.

- Coquand and Paulin "Inductively defined types" (COLOG-88) (inductive types, implementation had inductive families).
- Paulin-Mohring "Inductive definitions in the system Coq rules and properties" (TLCA 93) (inductive families)

Set-theoretic model, strong normalization proof

Rules for inductive types (expressed diagrammatically)

Let $\Phi : * \to *$ be a strictly positive operator. Then we can form $A = \mu\Phi$, intro, and rec such that

$$\begin{array}{ccc}
\Phi A & \xrightarrow{\ \mathrm{intro}\ } & A \\
\Big\downarrow {\scriptstyle \Phi\,\langle \mathrm{id}, \mathrm{rec}\, d \rangle} & & \Big\downarrow {\scriptstyle \langle \mathrm{id}, \mathrm{rec}\, d \rangle} \\
\Phi(\Sigma\, A\, C) & \xrightarrow[\ d\ ]{} & \Sigma\, A\, C
\end{array}$$

commutes.

This can be generalized to inductive families.

Uniform parametrization and the Paulin identity type

To recover usual rules for type formers we need to introduce the idea of *uniform parameters*. For example, $A, B : *$ are parameters in $A + B$ and $A \times B$.
Martin-Löf's identity type (in Agda). One parameter, two indices:

```
data I {A : Set} : A -> A -> Set where
  r : (a : A) -> I a a
```

Paulin's identity type in Agda (fix one argument a, two parameters, one index.)

```
data I {A : Set} (a : A) : A -> Set where
  r : I a a
```

Identity elimination

Martin-Löf:

```
J :  {A : Set}                              -- parameter
  -> {C : (x y : A) -> I x y -> Set}        -- induction form
  -> ((x : A) -> C x x (r x))               -- closure condit
  -> (a b : A) -> (c : I a b) -> C a b c    -- conclusion
J d .b b (r .b) = d b
```

Paulin:

```
J :  {A : Set} -> (a : A)                   -- parameters
  -> (C : (y : A) -> I a y -> Set)          -- induction forr
  -> C a r                                  -- closure condit
  -> (b : A) -> (c : I a b) -> C b c        -- conclusion
J .b d b r = d
```

The Swedish (predicative) point of view

- Intuitionistic type theory is an open system, we can add new inductive types when there is a need for it

The Swedish (predicative) point of view

- Intuitionistic type theory is an open system, we can add new inductive types when there is a need for it
- Can we add inductive families?

The Swedish (predicative) point of view

- Intuitionistic type theory is an open system, we can add new inductive types when there is a need for it
- Can we add inductive families?
- Is there a general formulation?

Martin-Löf 1972 on schema for 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 $(\Sigma : A)B(x)$, $A + B$, N_n and N as special cases. See Martin-Löf 1971 for a general formulation of inductive definitions in the language of ordinary first order predicate logic."

Martin-Löf 1984: "We can follow the same pattern used to define natural numbers to introduce other inductively defined sets. We see here the example of lists".

Extending ITT with inductive definitions (ID)

Add ID as $\mu X.\Phi$.
- Feferman (predicate logic)
- Constable and Mendler 1985 (inductive types)

Schema for ID.
- Martin-Löf 1971 (predicate logic)
- Backhouse 1986 (inductive types)
- Dybjer 1989 (inductive families)

Encode ID in W.
- Dybjer 1987 (inductive types)
- Petersson-Synek 1989 (general tree type)

Universe of codes for ID.
- Dybjer and Setzer 1999 (inductive-recursive types)
- Dybjer and Setzer 2002 (inductive-recursive families)

Extending ITT with inductive definitions (ID)

Add ID as $\mu X.\Phi$.
- Feferman (predicate logic)
- Constable and Mendler 1985 (inductive types)

Schema for ID.
- Martin-Löf 1971 (predicate logic)
- Backhouse 1986 (inductive types)
- Dybjer 1989 (inductive families)

Encode ID in W.
- Dybjer 1987 (inductive types)
- Petersson-Synek 1989 (general tree type)

Universe of codes for ID.
- Dybjer and Setzer 1999 (inductive-recursive types)
- Dybjer and Setzer 2002 (inductive-recursive families)

Does it matter whether we work in a predicative (ITT) or impredicative (CIC) setting?

Have we got the right formulation of inductive families?

Predicative (ITT+ID) vs impredicative (CIC) point of view?
Semantic foundation?

- Must index sets be small?
- Must index sets have decidable equality?

Beyond inductive definitions

- higher universes: Palmgren's super universe and universe hierarchies, Setzer's Mahlo universe
- inductive-recursive definitions
- universe of codes for inductive-recursive definitions (restricts to new formulation of inductive definitions)
- even higher universes: Setzer's autonomous Mahlo and Π_3-reflecting universes
- inductive-inductive definitions

Also pattern matching, termination checking, sized types in Agda
... is there some nice structure?
Coinductive types? Setzer 2011 has meaning explanations (first explicit attempt, cf Martin-Löf mathematics of infinity)