For (finitary) induction-induction, induction is enough

TYPES 2019, Kaposi-Kovács-Lafont

Monday talk, June 2021

Motivation for Inductive-inductive types

A foundational issue

Type theory should eat itself, Chapman '09

Type theory = **foundations** of mathematics
= **metatheory** for type theory ?

- Study type theory in type theory?
- Bootstrap Coq ?

Type theory as an inductive-inductive type [Chapman '09]

Definition **internal** to type theory

TYPES 2019 with Ambrus Kaposi and András Kovács

Construction of *inductive-inductive types with inductive types* (Agda)

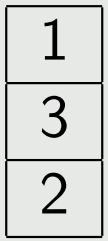
Examples of inductive types in type theory

Simple inductive types: lists

$\texttt{Inductive}\ \mathit{list} :=$

$\quad \mid \mathit{nil} \quad : \mathit{list}$

$\quad \mid \mathit{cons} : \mathbb{N} \times \mathit{list} \to \mathit{list}$



$= \mathit{cons}(1, \mathit{cons}(3, \mathit{cons}(2, \mathit{nil})))$

Mutual inductives types: odd/even size lists

$\texttt{Inductive}\ \mathit{list}_{even} :=$

$\quad \mid \mathit{nil} \quad : \mathit{list}_{even}$

$\quad \mid \mathit{econs} : \mathbb{N} \times \mathit{list}_{odd} \to \mathit{list}_{even}$

$\texttt{with}\ \mathit{list}_{odd} :=$

$\quad \mid \mathit{ocons} : \mathbb{N} \times \mathit{list}_{even} \to \mathit{list}_{odd}$

Indexed inductive types: fixed size lists (dependent types)

$$\boxed{\mathit{vector}(n) = \text{type of lists of size } n}$$

$\texttt{Inductive}\ \mathit{vector} : \mathbb{N} \to \texttt{Type} :=$

$\quad \mid \mathit{nil} \quad : \mathit{vector}(0)$

$\quad \mid \mathit{cons} : \mathbb{N} \times \mathit{vector}(n) \to \mathit{vector}(1+n)$

Inductive-inductive types

Inductif-inductive ≈ *generalised* mutual indexed inductive

$$\texttt{Inductive}\ A := \ldots\ \texttt{with}\ B : \underline{A \to \texttt{Type}} := \ldots$$

New feature : the type of B depends on A
(not allowed for an indexed mutual inductive)

Exemple : type theory !

$$\begin{array}{ll} \texttt{Inductive}\ \textbf{\textit{context}} := \ldots & \Gamma \vdash \\ \quad \texttt{with}\ \textit{types} : \underline{\textbf{\textit{context}} \to \texttt{Type}} := \ldots & \Gamma \vdash A \end{array}$$

Conjecture : *Inductive-inductive definitions*, PhD [Forsberg 2013]

inductive-inductive $>$ inductive (in *extensional* type theory)

TYPES'19 : construction of ***finitary*** inductive-inductive types

Conjecture invalidated in the *finitary case* (finite arities)

Constructing finitary inductive-inductive types

[Kaposi-Kovács 2019]

Definition of a type theory $\mathbb{T}$, such that:

$$\begin{array}{ccc} \text{construction of } \mathbb{T} & \Rightarrow & \exists \text{ inductive-inductive} \\ \text{(with its recurrence principle)} & & \text{(finitary case)} \end{array}$$

My contribution : construction[1] of $\mathbb{T}$ (Agda, 6000 LoC)

Major difficulty: recurrence principle for $\mathbb{T}$

$\Rightarrow$ [Kaposi-Kovács-Lafont TYPES 2019]

[1] `https://github.com/amblafont/UniversalII/tree/cwf-syntax`

Recurrence principle $\approx$ Initiality

Example : natural numbers

Induction	$\dfrac{P(0) \quad P(n) \Rightarrow P(n+1)}{\forall n.P(n)}$
Recursion	$\exists\, f : \mathbb{N} \to X$ $\begin{cases} f(0) = f_0 \\ f(n+1) = \ldots f(n) \ldots \end{cases}$
Initiality	"unique" recursion $\ldots \exists! f : \mathbb{N} \to X \ldots$

$$\boxed{\text{Initiality} \Leftrightarrow \text{Induction} + \text{Recursion}}$$

Recurrence principle ≈ Initiality

Example : natural numbers

Induction	$\dfrac{P(0) \quad P(n) \Rightarrow P(n+1)}{\forall n.P(n)}$
Recursion	$\exists\, f : \mathbb{N} \to X$ $\begin{cases} f(0) = f_0 \\ f(n+1) = \ldots f(n) \ldots \end{cases}$
Initiality	"unique" recursion $\ldots \exists! f : \mathbb{N} \to X \ldots$

$\boxed{\forall P} \Rightarrow$ "transport hell"

$\boxed{\text{Initiality} \Leftrightarrow \text{Induction} + \text{Recursion}}$

The « transport hell »

Explicit coercions in type theory

- ✓ Decidable type-checking
- × Proofs jammed by equality witnesses

A relief : turning equalities into rewrite rules (Agda)

- rev_append: $n' + (1 + m) \longrightarrow 1 + n' + m$: no coercion needed
- experimental feature exploited in my formalisation

Conclusion

Inductive-inductive types

- a use case: define type theory **in** type theory
- constructible from indexed inductive types

From transport hell to homotopy heaven

× Dependent types are tedious (no transport hell in Isabelle)

⇒ limit their use

- CompCert (Leroy & co)
- Felt-Thomson, 4-colour theorems (Gonthier & co)
- big Coq formalisations

but...

✓ This complexity can be used to model topological spaces

The simplicial model of univalent foundations,

Voevodsky & co, 2012

⇒ Homotopy type theory

Homotopy type theory

2009

Univalent Axiom (Voevodsky, Fields 2002)

isomorphism $\Leftrightarrow$ *equality*

2012-2013 Special year (IAS, Princeton)
Univalent foundations for mathematics

Synthetic homotopy (types = topological spaces)

Some examples (e.g., Hopf fibration) in

[Cubical synthetic homotopy theory, Mörtberg-Pujet, CPP'20]

Merci pour votre attention.

Travaux connexes : réduction des inductifs-inductifs aux inductifs

- *Inductive-inductive definitions*, [Forsberg 2013]
  Thèse de doctorat
- *Constructing Inductive-Inductive Types in Cubical Type Theory*, [Hugunin 2019], sur un exemple

✓ Construction directe, sans introduire $\mathbb{T}$

× Principe de récurrence partiel

Perspectives

- Adapter aux inductifs-inductifs infinitaires (W-type, Acc)
- Adapter la construction aux inductifs-inductifs quotients
- Adapter la construction à la théorie des types intensionnelle ?