EFOP-3.6.2-16-2017-00013

European Union

Quotient inductive-inductive types

Ambrus Kaposi (ELTE)
j.w.w. András Kovács (ELTE) & Thorsten Altenkirch (Nottingham)

Conference on Software Technology and Cyber Security (STCS)
22 February 2019

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social Fund

INVESTING IN YOUR FUTURE

Overview

Inductive types by examples

- Universal inductive type

Indexed inductive types by examples

- Universal indexed inductive type

Quotient inductive types (QITs) by examples

- UNIVERSAL QIT

Inductive types

are specified by their constructors.

E.g.

$$\begin{aligned} \mathsf{Bool} &: \mathsf{Type} \\ \mathsf{true} &: \mathsf{Bool} \\ \mathsf{false} &: \mathsf{Bool} \end{aligned}$$

means

$$\mathsf{Bool} = \{\mathsf{true}, \mathsf{false}\}.$$

Another example

$$\begin{aligned} \mathbb{N} &: \mathsf{Type} \\ \mathsf{zero} &: \mathbb{N} \\ \mathsf{suc} &: \mathbb{N} \to \mathbb{N} \end{aligned}$$

means

$$\mathbb{N} = \{\mathsf{zero},\ \mathsf{suc}\,\mathsf{zero},\ \mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero}),\ \mathsf{suc}\,(\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})),\ \ldots\},$$

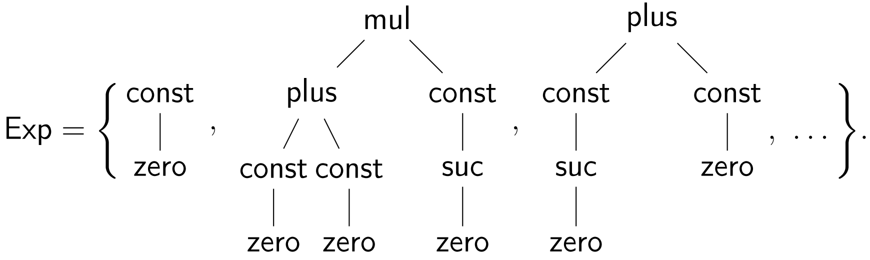
usually written

$$\mathbb{N} = \{0,\ 1,\ 2,\ \ldots\}.$$

Another example

$$
\begin{aligned}
\mathsf{Exp} &: \mathsf{Type} \\
\mathsf{const} &: \mathbb{N} \to \mathsf{Exp} \\
\mathsf{plus} &: \mathsf{Exp} \to \mathsf{Exp} \to \mathsf{Exp} \\
\mathsf{mul} &: \mathsf{Exp} \to \mathsf{Exp} \to \mathsf{Exp}
\end{aligned}
$$

means



Another example

$$\begin{aligned} \mathsf{Exp} &: \mathsf{Type} \\ \mathsf{const} &: \mathbb{N} \to \mathsf{Exp} \\ \mathsf{plus} &: \mathsf{Exp} \to \mathsf{Exp} \to \mathsf{Exp} \\ \mathsf{mul} &: \mathsf{Exp} \to \mathsf{Exp} \to \mathsf{Exp} \end{aligned}$$

written in a linear notation as

$$\begin{aligned} &\mathsf{Exp} = \\ &\;\Big\{\mathsf{const\,zero}, \\ &\;\;\mathsf{mul}\,\big(\mathsf{plus}\,(\mathsf{const}\,(\mathsf{suc\,zero}))\,(\mathsf{const}\,(\mathsf{suc\,zero}))\big)\,\big(\mathsf{const}\,(\mathsf{suc\,zero})\big), \\ &\;\;\mathsf{plus}\,\big(\mathsf{const}\,(\mathsf{suc\,zero})\big)\,(\mathsf{const\,zero}),\;\ldots\Big\}. \end{aligned}$$

Another example

$$\begin{aligned} \mathbb{N}' &: \mathsf{Type} \\ \mathsf{suc} &: \mathbb{N}' \to \mathbb{N}' \end{aligned}$$

means

$$\mathbb{N}' = \{\}.$$

Why *inductive*? We can do induction!

On Bool: $(P : \mathsf{Bool} \to \mathsf{Type}) \to P\,\mathsf{true} \to P\,\mathsf{false} \to$
$(b : \mathsf{Bool}) \to P\,b$

On $\mathbb{N}$: $(P : \mathbb{N} \to \mathsf{Type}) \to P\,\mathsf{zero} \to$
$((n : \mathbb{N}) \to P\,n \to P\,(\mathsf{suc}\,n)) \to (n : \mathbb{N}) \to P\,n$

On Exp: $(P : \mathsf{Exp} \to \mathsf{Type}) \to ((n : \mathbb{N}) \to P\,(\mathsf{const}\,n)) \to$
$((e\,e' : \mathsf{Exp}) \to P\,e \to P\,e' \to P\,(\mathsf{plus}\,e\,e')) \to$
$((e\,e' : \mathsf{Exp}) \to P\,e \to P\,e' \to P\,(\mathsf{mul}\,e\,e')) \to$
$(e : \mathsf{Exp}) \to P\,e$

Not an inductive type

$$\begin{aligned} \mathsf{Neg} &: \mathsf{Type} \\ \mathsf{con} &: (\mathsf{Neg} \to \bot) \to \mathsf{Neg} \end{aligned}$$

The induction principle:

$$\begin{aligned} \mathsf{elimNeg} : {} & (P : \mathsf{Neg} \to \mathsf{Type}) \to ((f : \mathsf{Neg} \to \bot) \to P\,(\mathsf{con}\, f)) \to \\ & (n : \mathsf{Neg}) \to P\, n \end{aligned}$$

Now we can do something bad:

$$\begin{aligned} \mathsf{probl} &: \mathsf{Neg} \to \bot := \lambda n.\mathsf{elimNeg}\,(\lambda\_.\mathsf{Neg} \to \bot)\,(\lambda f.f)\; n\; n \\ \mathsf{PROBL} &: \bot \qquad\qquad := \mathsf{probl}\,(\mathsf{con}\;\mathsf{probl}) \end{aligned}$$

What is a generic definition?

We have $\bot$, $\top$, $+$ and $\times$ types.

Universal inductive type (Martin-Löf, 1984): for every

$$S : \mathrm{Type} \qquad \text{and} \qquad P : S \to \mathrm{Type}$$

there is an inductive type

$$\begin{aligned} \mathrm{W} &: \mathrm{Type} \\ \mathrm{sup} &: (s : S) \to (P\,s \to \mathrm{W}) \to \mathrm{W} \end{aligned}$$

E.g. $\mathbb{N}$ is given by

$$S := \top + \top \qquad P\,(\mathrm{inl}\,\mathrm{tt}) := \bot \qquad P\,(\mathrm{inr}\,\mathrm{tt}) := \top.$$

An indexed inductive type

$$\begin{aligned}
\mathsf{Vec} &: \mathbb{N} \to \mathsf{Type} \\
\mathsf{nil} &: \mathsf{Vec}\,\mathsf{zero} \\
\mathsf{cons} &: (n : \mathbb{N}) \to \mathsf{Bool} \to \mathsf{Vec}\, n \to \mathsf{Vec}\,(\mathsf{suc}\, n)
\end{aligned}$$

means

$$\begin{aligned}
\mathsf{Vec}\,\mathsf{zero} &= \{\mathsf{nil}\} \\
\mathsf{Vec}\,(\mathsf{suc}\,\mathsf{zero}) &= \{\mathsf{cons}\,\mathsf{zero}\,\mathsf{true}\,\mathsf{nil},\ \mathsf{cons}\,\mathsf{zero}\,\mathsf{false}\,\mathsf{nil}\} \\
\mathsf{Vec}\,(\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})) &= \{\mathsf{cons}\,(\mathsf{suc}\,\mathsf{zero})\,\mathsf{true}\,(\mathsf{cons}\,\mathsf{zero}\,\mathsf{true}\,\mathsf{nil}),\ \ldots\} \\
\ldots
\end{aligned}$$

An indexed inductive type

$$\begin{aligned}
\mathsf{Vec} &: \mathbb{N} \to \mathsf{Type} \\
\mathsf{nil} &: \mathsf{Vec}\,\mathsf{zero} \\
\mathsf{cons} &: (n : \mathbb{N}) \to \mathsf{Bool} \to \mathsf{Vec}\,n \to \mathsf{Vec}\,(\mathsf{suc}\,n)
\end{aligned}$$

usually written as

$$\begin{aligned}
\mathsf{Vec}\,\mathsf{zero} &= \{[]\} \\
\mathsf{Vec}\,(\mathsf{suc}\,\mathsf{zero}) &= \{[\mathsf{true}],\ [\mathsf{false}]\} \\
\mathsf{Vec}\,(\mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero})) &= \{[\mathsf{true},\mathsf{true}],\ [\mathsf{true},\mathsf{false}],\ [\mathsf{false},\mathsf{true}],\ \ldots\} \\
\ldots
\end{aligned}$$

A mutual inductive type

$$
\begin{aligned}
\mathsf{Cmd} &: \mathsf{Type} \\
\mathsf{Block} &: \mathsf{Type} \\
\mathsf{skip} &: \mathsf{Cmd} \\
\mathsf{ifelse} &: \mathsf{Exp} \to \mathsf{Block} \to \mathsf{Block} \to \mathsf{Cmd} \\
\mathsf{assign} &: \mathbb{N} \to \mathsf{Exp} \to \mathsf{Cmd} \\
\mathsf{single} &: \mathsf{Cmd} \to \mathsf{Block} \\
\mathsf{semicol} &: \mathsf{Cmd} \to \mathsf{Block} \to \mathsf{Block}
\end{aligned}
$$

BNF definitions are usually mutual inductive types.

Universal indexed/mutual inductive type

Mutual inductive types can be reduced to indexed ones.

$$\text{Cmd, Block} \qquad \text{becomes} \qquad \text{CmdOrBlock} : \text{Bool} \to \text{Type}$$

Altenkirch–Ghani–Hancock–McBride, 2015: for every

$$S : \text{Type} \qquad \text{and} \qquad P : S \to \text{Type} \qquad \text{and}$$

$$\mathit{out} : S \to I \qquad \text{and} \qquad \mathit{in} : (s : S) \to P\,s \to I$$

there is the indexed inductive type

$$\begin{aligned} \mathrm{W} \quad &: I \to \text{Type} \\ \sup &: (s : S)\big((p : P\,s) \to \mathrm{W}\,(\mathit{in}\,s\,p)\big) \to \mathrm{W}\,(\mathit{out}\,s) \end{aligned}$$

Integers

$$\begin{aligned} \mathbb{Z} &: \text{Type} \\ \text{pair} &: \mathbb{N} \to \mathbb{N} \to \mathbb{Z} \\ \text{quot} &: (a\, b\, a'\, b' : \mathbb{N}) \to a + b' = a' + b \to \text{pair}\, a\, b = \text{pair}\, a'\, b' \end{aligned}$$

means

$$\begin{aligned} \mathbb{Z} = \{ & \{\text{pair}\,0\,0,\ \text{pair}\,1\,1,\ \text{pair}\,2\,2,\ \ldots\}, \\ & \{\text{pair}\,0\,1,\ \text{pair}\,1\,2,\ \text{pair}\,2\,3,\ \ldots\}, \\ & \{\text{pair}\,1\,0,\ \text{pair}\,2\,1,\ \text{pair}\,3\,2,\ \ldots\}, \\ & \{\text{pair}\,0\,2,\ \text{pair}\,1\,3,\ \text{pair}\,2\,4,\ \ldots\}, \\ & \ldots \} \end{aligned}$$

Quotients

Given $A : \mathrm{Type}$, $R : A \to A \to \mathrm{Type}$, the quotient type is

$$
\begin{aligned}
A/R &: \mathrm{Type} \\
[-] &: A \to A/R \\
\mathrm{quot} &: (a\, a' : A) \to R\, a\, a' \to [a] = [a']
\end{aligned}
$$

Cauchy Real numbers

$$
\begin{array}{ll}
\mathbb{R} & : \mathsf{Type} \\
\mathsf{P} & : \mathbb{Q}_+ \to \mathbb{R} \to \mathbb{R} \to \mathsf{Type} \\
\mathsf{rat} & : \mathbb{Q} \to \mathbb{R} \\
\mathsf{lim} & : (f : \mathbb{Q}_+ \to \mathbb{R}) \to \big((\delta\,\epsilon : \mathbb{Q}_+) \to \mathsf{P}\,(\delta + \epsilon)\,(f\,\delta)\,(f\,\epsilon)\big) \to \mathbb{R} \\
\mathsf{eq} & : (u\,v : \mathbb{R}) \to \big((\epsilon : \mathbb{Q}_+) \to \mathsf{P}\,\epsilon\,u\,v\big) \to u = v \\
\mathsf{ratrat} & : (q\,r : \mathbb{Q})(\epsilon : \mathbb{Q}_+)(-\epsilon < q - r < \epsilon) \to \mathsf{P}\,\epsilon\,(\mathsf{rat}\,q)\,(\mathsf{rat}\,r) \\
\mathsf{ratlim} & : \mathsf{P}\,(\epsilon - \delta)\,(\mathsf{rat}\,q)\,(g\,\delta) \to \mathsf{P}\,\epsilon\,(\mathsf{rat}\,q)\,(\mathsf{lim}\,g) \\
\mathsf{limrat} & : \mathsf{P}\,(\epsilon - \delta)\,(f\,\delta)\,(\mathsf{rat}\,r) \to \mathsf{P}\,\epsilon\,(\mathsf{lim}\,f)\,(\mathsf{rat}\,r) \\
\mathsf{limlim} & : \mathsf{P}\,(\epsilon - \delta - \eta)\,(f\,\delta)\,(g\,\eta) \to \mathsf{P}\,\epsilon\,(\mathsf{lim}\,f)\,(\mathsf{lim}\,g) \\
\mathsf{trunc} & : (\xi\,\zeta : \mathsf{P}\,\epsilon\,u\,v) \to \xi = \zeta
\end{array}
$$

(Homotopy Type Theory book, 2013)

Partiality monad for non-terminating programs

(Altenkirch–Danielsson–Kraus, 2017)

$$
\begin{array}{ll}
A_\bot & : \mathsf{Type} \\
- \sqsubseteq - & : A_\bot \to A_\bot \to \mathsf{Type} \\
\eta & : A \to A_\bot \\
\bot & : A_\bot \\
\bigsqcup & : (f : \mathbb{N} \to A_\bot)\big((n : \mathbb{N}) \to f\, n \sqsubseteq f\,(n+1)\big) \to A_\bot \\
\mathsf{refl} & : d \sqsubseteq d \\
\mathsf{inf} & : \bot \sqsubseteq d \\
\mathsf{in} & : \big((n : \mathbb{N}) \to f\, n \sqsubseteq d\big) \to \bigsqcup f\, p \sqsubseteq d \\
\mathsf{out} & : \bigsqcup f\, p \sqsubseteq d \to (n : \mathbb{N}) \to f\, n \sqsubseteq d \\
\mathsf{antisym} & : (d\, d' : A_\bot) \to d \sqsubseteq d' \to d' \sqsubseteq d \to d = d' \\
\mathsf{trunc} & : (\xi\, \zeta : d \sqsubseteq d') \to \xi = \zeta
\end{array}
$$

Algebraic syntax for a programming language

$$
\begin{aligned}
&\mathsf{Ty} &&: \mathsf{Type}\\
&\mathsf{Tm} &&: \mathsf{Ty} \to \mathsf{Type}\\
&\mathsf{Bool},\ \mathsf{Nat} &&: \mathsf{Ty}\\
&\mathsf{true},\ \mathsf{false} &&: \mathsf{Tm}\,\mathsf{Bool}\\
&\mathsf{if}{-}\mathsf{then}{-}\mathsf{else}{-} &&: \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A\\
&\mathsf{num} &&: \mathbb{N} \to \mathsf{Tm}\,\mathsf{Nat}\\
&\mathsf{isZero} &&: \mathsf{Tm}\,\mathsf{Nat} \to \mathsf{Tm}\,\mathsf{Bool}\\
&\mathsf{if}\beta_1 &&: \mathsf{if}\,\mathsf{true}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t\\
&\mathsf{if}\beta_2 &&: \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t'\\
&\mathsf{isZero}\beta_1 &&: \mathsf{isZero}\,(\mathsf{num}\,0) = \mathsf{true}\\
&\mathsf{isZero}\beta_2 &&: \mathsf{isZero}\,(\mathsf{num}\,(1+n)) = \mathsf{false}
\end{aligned}
$$

(Altenkirch–Kaposi, 2016)

A domain-specific language for QIT signatures

$$\frac{}{\vdash \cdot} \qquad \frac{\Gamma \vdash A}{\vdash \Gamma, x : A} \qquad \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathsf{U}} \qquad \frac{\Gamma \vdash a : \mathsf{U}}{\Gamma \vdash \underline{a}}$$

$$\frac{\Gamma \vdash a : \mathsf{U} \qquad \Gamma, x : \underline{a} \vdash B}{\Gamma \vdash (x : a) \Rightarrow B} \qquad \frac{\Gamma \vdash t : (x : a) \Rightarrow B \qquad \Gamma \vdash u : \underline{a}}{\Gamma \vdash t \mathbin{@} u : B[x \mapsto u]}$$

$$\frac{\Gamma \vdash u : \underline{a} \qquad \Gamma \vdash v : \underline{a}}{\Gamma \vdash u = v} \qquad \dots$$

A signature is a context Γ, e.g.

$$(\cdot,\ N : \mathsf{U},\ zero : \underline{N},\ suc : N \Rightarrow \underline{N})$$

$$(\cdot,\ Ty : \mathsf{U},\ Tm : Ty \Rightarrow \mathsf{U},\ Bool : \underline{Ty},\ true : \underline{Tm \mathbin{@} Bool},\ \dots)$$

This is a QIT itself

$$
\begin{aligned}
&\mathsf{Con} && : \mathsf{Type} \\
&\mathsf{Ty} && : \mathsf{Con} \to \mathsf{Type} \\
&\mathsf{Var} && : \mathsf{Con} \to \mathsf{Type} \\
&\mathsf{Tm} && : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Type} \\
&\cdot && : \mathsf{Con} \\
&(-,- : -) && : (\Gamma : \mathsf{Con}) \to \mathsf{Var}\,\Gamma \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
&\mathsf{U} && : \mathsf{Ty}\,\Gamma \\
&\underline{-} && : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma \\
&(- : -) \Rightarrow - && : \mathsf{Var}\,\Gamma \to (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,(\Gamma, x : \underline{a}) \to \mathsf{Ty}\,\Gamma \\
&- \mathbin{@} - && : \mathsf{Tm}\,\Gamma\,((x : a) \Rightarrow B) \to (u : \mathsf{Tm}\,\Gamma\,\underline{a}) \to \\
& && \quad \mathsf{Tm}\,\Gamma\,(B[x \mapsto u])
\end{aligned}
$$

...

Results

- A generic definition of signatures for QITs which includes all the known examples
- Description of the induction principle
  - Kaposi–Kovács, FSCD 2018
- If the universal QIT exists, then all of them exist
  - Kaposi–Kovács–Altenkirch, POPL 2019
- Existence of the universal QIT
  - People proved this in different settings, e.g. Brunerie
  - Part without quotients done (by Ambroise Lafont), full version further work

EFOP-3.6.2-16-2017-00013

European Union

THANK YOU FOR YOUR ATTENTION!

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social Fund

INVESTING IN YOUR FUTURE