EFOP-3.6.2-16-2017-00013

European Union

Quotient inductive-inductive types

Ambrus Kaposi   András Kovács

Eötvös Loránd University

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

Plan

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

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

Another example

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

means

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

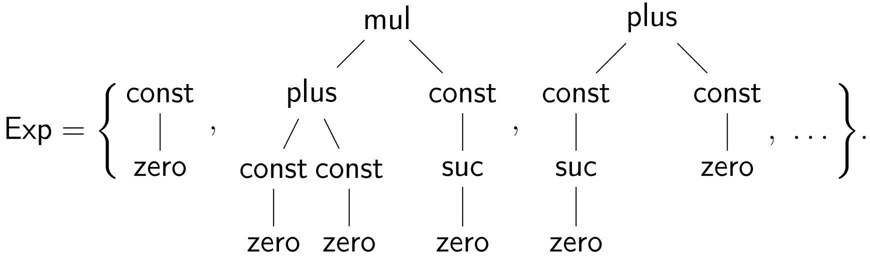
usually written

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

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

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

usually written as

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

Another example

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

means

Another example

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

means

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

Why *inductive*?

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

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

Not an inductive type

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

The eliminator:

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

Not an inductive type

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

The eliminator:

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

Now we can do something bad:

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

Plan

Inductive types by examples

- Universal inductive type

Indexed inductive types by examples

- Universal indexed inductive type

Quotient inductive types (QITs) by examples

- UNIVERSAL QIT

What is a generic definition?

We have $\bot$, $\top$, $+$ and $\times$ types.
Universal inductive type (Per Martin-Löf, 1984): for every

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

there is an inductive type

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

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

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

Plan

Inductive types by examples

- Universal inductive type

Indexed inductive types by examples

- Universal indexed inductive type

Quotient inductive types (QITs) by examples

- UNIVERSAL QIT

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

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

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.

Plan

Inductive types by examples

Universal inductive type

Indexed inductive types by examples

Universal indexed inductive type

Quotient inductive types (QITs) by examples

UNIVERSAL QIT

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

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

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

there is the indexed inductive type

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

Plan

Inductive types by examples

Universal inductive type

Indexed inductive types by examples

Universal indexed inductive type

Quotient inductive types (QITs) by examples

UNIVERSAL QIT

Integers

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

means

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

Quotients

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

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

Cauchy Real numbers

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

Partiality monad for non-terminating programs

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

Algebraic syntax for a programming language

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

Plan

Inductive types by examples

- Universal inductive type

Indexed inductive types by examples

- Universal indexed inductive type

Quotient inductive types (QITs) by examples

- UNIVERSAL QIT

A domain-specific language for QIT signatures

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

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

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

A domain-specific language for QIT signatures

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

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

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

A signature is a context Γ, e.g.

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

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

This is a QIT itself

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

$\ldots$

Results

- A generic definition of signatures for QITs which includes all the known examples
- Description of eliminator
- If the universal QIT exists, then all of them exist
  - People proved this in different settings
  - We plan to do the explicit construction

EFOP-3.6.2-16-2017-00013

innovation
informatics
infocommunications
ELTE PPKE BME

European Union

THANK YOU FOR YOUR ATTENTION!

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social
Fund

INVESTING IN YOUR FUTURE