

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



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.

`Bool : Type`

`true : Bool`

`false : Bool`

means

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

## Another example

$\mathbb{N} : \text{Type}$

$\text{zero} : \mathbb{N}$

$\text{suc} : \mathbb{N} \rightarrow \mathbb{N}$

means

$\mathbb{N} = \{\text{zero}, \text{suc zero}, \text{suc}(\text{suc zero}), \text{suc}(\text{suc}(\text{suc zero})), \dots\},$

## Another example

```
 $\mathbb{N} : \text{Type}$   
 $\text{zero} : \mathbb{N}$   
 $\text{suc} : \mathbb{N} \rightarrow \mathbb{N}$ 
```

means

$\mathbb{N} = \{\text{zero}, \text{suc zero}, \text{suc}(\text{suc zero}), \text{suc}(\text{suc}(\text{suc zero})), \dots\},$

usually written

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

# Another example

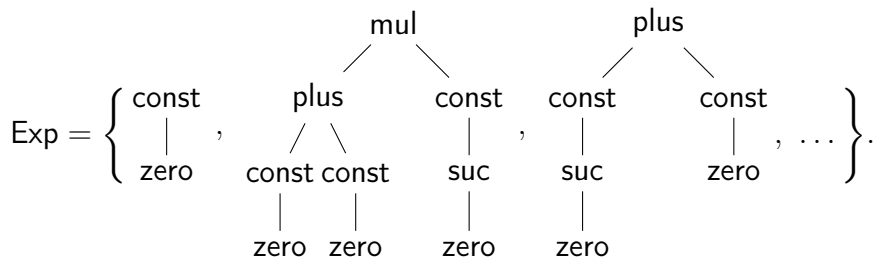
$\text{Exp} : \text{Type}$

$\text{const} : \mathbb{N} \rightarrow \text{Exp}$

$\text{plus} : \text{Exp} \rightarrow \text{Exp} \rightarrow \text{Exp}$

$\text{mul} : \text{Exp} \rightarrow \text{Exp} \rightarrow \text{Exp}$

means



# Another example

$\text{Exp} : \text{Type}$   
 $\text{const} : \mathbb{N} \rightarrow \text{Exp}$   
 $\text{plus} : \text{Exp} \rightarrow \text{Exp} \rightarrow \text{Exp}$   
 $\text{mul} : \text{Exp} \rightarrow \text{Exp} \rightarrow \text{Exp}$

usually written as

$\text{Exp} =$   
 $\left\{ \begin{array}{l} \text{const zero}, \\ \text{mul} \left( \text{plus} \left( \text{const} \left( \text{suc zero} \right) \right) \left( \text{const} \left( \text{suc zero} \right) \right) \right) \left( \text{const} \left( \text{suc zero} \right) \right), \\ \text{plus} \left( \text{const} \left( \text{suc zero} \right) \right) \left( \text{const zero} \right), \dots \end{array} \right\}.$



## Another example

$$\begin{aligned} \mathbb{N}' &: \text{Type} \\ \text{suc} &: \mathbb{N}' \rightarrow \mathbb{N}' \end{aligned}$$

means

## Another example

$\mathbb{N}' : \text{Type}$

$\text{suc} : \mathbb{N}' \rightarrow \mathbb{N}'$

means

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

# Why *inductive*?

# Why *inductive*? We can do induction!

On Bool:  $(P : \text{Bool} \rightarrow \text{Type}) \rightarrow P \text{ true} \rightarrow P \text{ false} \rightarrow$   
 $(b : \text{Bool}) \rightarrow P b$

On  $\mathbb{N}$ :  $(P : \mathbb{N} \rightarrow \text{Type}) \rightarrow P \text{ zero} \rightarrow$   
 $((n : \mathbb{N}) \rightarrow P n \rightarrow P (\text{suc } n)) \rightarrow (n : \mathbb{N}) \rightarrow P n$

On Exp:  $(P : \text{Exp} \rightarrow \text{Type}) \rightarrow ((n : \mathbb{N}) \rightarrow P (\text{const } n)) \rightarrow$   
 $((e \ e' : \text{Exp}) \rightarrow P e \rightarrow P e' \rightarrow P (\text{plus } e \ e')) \rightarrow$   
 $((e \ e' : \text{Exp}) \rightarrow P e \rightarrow P e' \rightarrow P (\text{mul } e \ e')) \rightarrow$   
 $(e : \text{Exp}) \rightarrow P e$

# Not an inductive type

Neg : Type

con : (Neg  $\rightarrow$   $\perp$ )  $\rightarrow$  Neg

# Not an inductive type

$\text{Neg} : \text{Type}$

$\text{con} : (\text{Neg} \rightarrow \perp) \rightarrow \text{Neg}$

The induction principle:

$\text{elimNeg} : (P : \text{Neg} \rightarrow \text{Type}) \rightarrow ((f : \text{Neg} \rightarrow \perp) \rightarrow P(\text{con } f)) \rightarrow$   
 $(n : \text{Neg}) \rightarrow P\ n$

# Not an inductive type

$\text{Neg} : \text{Type}$

$\text{con} : (\text{Neg} \rightarrow \perp) \rightarrow \text{Neg}$

The induction principle:

$\text{elimNeg} : (P : \text{Neg} \rightarrow \text{Type}) \rightarrow ((f : \text{Neg} \rightarrow \perp) \rightarrow P(\text{con } f)) \rightarrow$   
 $(n : \text{Neg}) \rightarrow P\ n$

Now we can do something bad:

$\text{probl} : \text{Neg} \rightarrow \perp := \lambda n. \text{elimNeg} (\lambda \_ . \text{Neg} \rightarrow \perp) (\lambda f. f) n\ n$   
 $\text{PROBL} : \perp := \text{probl} (\text{con probl})$

# 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  $\perp$ ,  $\top$ ,  $+$  and  $\times$  types.

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

$$S : \text{Type} \quad \text{and} \quad P : S \rightarrow \text{Type}$$

there is an inductive type

$$W : \text{Type}$$

$$\text{sup} : (s : S) \rightarrow (P s \rightarrow W) \rightarrow W$$

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

$$S := \top + \top \quad P(\text{inl } \text{tt}) := \perp \quad P(\text{inr } \text{tt}) := \top.$$

# 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

$\text{Vec} : \mathbb{N} \rightarrow \text{Type}$

$\text{nil} : \text{Vec zero}$

$\text{cons} : (n : \mathbb{N}) \rightarrow \text{Bool} \rightarrow \text{Vec } n \rightarrow \text{Vec (suc } n)$

means

$\text{Vec zero} = \{\text{nil}\}$

$\text{Vec (suc zero)} = \{\text{cons zero true nil}, \text{cons zero false nil}\}$

$\text{Vec (suc (suc zero))} = \{\text{cons (suc zero) true (cons zero true nil)}, \dots\}$

...

# An indexed inductive type

$\text{Vec} : \mathbb{N} \rightarrow \text{Type}$

$\text{nil} : \text{Vec zero}$

$\text{cons} : (n : \mathbb{N}) \rightarrow \text{Bool} \rightarrow \text{Vec } n \rightarrow \text{Vec } (\text{suc } n)$

usually written as

$\text{Vec zero} = \{[]\}$

$\text{Vec } (\text{suc zero}) = \{[\text{true}], [\text{false}]\}$

$\text{Vec } (\text{suc } (\text{suc zero})) = \{[\text{true}, \text{true}], [\text{true}, \text{false}], [\text{false}, \text{true}], \dots\}$

...

# A mutual inductive type

Cmd : Type  
Block : Type  
skip : Cmd  
ifelse : Exp  $\rightarrow$  Block  $\rightarrow$  Block  $\rightarrow$  Cmd  
assign :  $\mathbb{N} \rightarrow$  Exp  $\rightarrow$  Cmd  
single : Cmd  $\rightarrow$  Block  
semicolon : Cmd  $\rightarrow$  Block  $\rightarrow$  Block

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.

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

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

$S : \text{Type}$  and  $P : S \rightarrow \text{Type}$  and

$\text{out} : S \rightarrow I$  and  $\text{in} : (s : S) \rightarrow P s \rightarrow I$

there is the indexed inductive type

$W : I \rightarrow \text{Type}$

$\text{sup} : (s : S) ((p : P s) \rightarrow W (\text{in } s p)) \rightarrow W (\text{out } s)$

# 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

$\mathbb{Z}$  : Type

pair :  $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{Z}$

quot :  $(a\ b\ a'\ b' : \mathbb{N}) \rightarrow a + b' = a' + b \rightarrow \text{pair } a\ b = \text{pair } a'\ b'$

means

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

# Quotients

Given  $A : \text{Type}$ ,  $R : A \rightarrow A \rightarrow \text{Type}$ , the quotient type is

$$A/R : \text{Type}$$
$$[-] : A \rightarrow A/R$$
$$\text{quot} : (a\ a' : A) \rightarrow R\ a\ a' \rightarrow [a] = [a']$$

# Cauchy Real numbers

$\mathbb{R}$  : Type

$P$  :  $\mathbb{Q}_+ \rightarrow \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{Type}$

$\text{rat}$  :  $\mathbb{Q} \rightarrow \mathbb{R}$

$\text{lim}$  :  $(f : \mathbb{Q}_+ \rightarrow \mathbb{R}) \rightarrow ((\delta \epsilon : \mathbb{Q}_+) \rightarrow P(\delta + \epsilon)(f \delta)(f \epsilon)) \rightarrow \mathbb{R}$

$\text{eq}$  :  $(u v : \mathbb{R}) \rightarrow ((\epsilon : \mathbb{Q}_+) \rightarrow P \epsilon u v) \rightarrow u = v$

$\text{ratrat}$  :  $(q r : \mathbb{Q})(\epsilon : \mathbb{Q}_+)(-\epsilon < q - r < \epsilon) \rightarrow P \epsilon (\text{rat } q) (\text{rat } r)$

$\text{ratlim}$  :  $P(\epsilon - \delta)(\text{rat } q)(g \delta) \rightarrow P \epsilon (\text{rat } q) (\text{lim } g)$

$\text{limrat}$  :  $P(\epsilon - \delta)(f \delta)(\text{rat } r) \rightarrow P \epsilon (\text{lim } f) (\text{rat } r)$

$\text{limlim}$  :  $P(\epsilon - \delta - \eta)(f \delta)(g \eta) \rightarrow P \epsilon (\text{lim } f) (\text{lim } g)$

$\text{trunc}$  :  $(\xi \zeta : P \epsilon u v) \rightarrow \xi = \zeta$

# Partiality monad for non-terminating programs

$A_{\perp} \quad : \text{Type}$

$- \sqsubseteq - \quad : A_{\perp} \rightarrow A_{\perp} \rightarrow \text{Type}$

$\eta \quad : A \rightarrow A_{\perp}$

$\perp \quad : A_{\perp}$

$\bigsqcup \quad : (f : \mathbb{N} \rightarrow A_{\perp})((n : \mathbb{N}) \rightarrow f\ n \sqsubseteq f\ (n + 1)) \rightarrow A_{\perp}$

$\text{refl} \quad : d \sqsubseteq d$

$\text{inf} \quad : \perp \sqsubseteq d$

$\text{in} \quad : ((n : \mathbb{N}) \rightarrow f\ n \sqsubseteq d) \rightarrow \bigsqcup f\ p \sqsubseteq d$

$\text{out} \quad : \bigsqcup f\ p \sqsubseteq d \rightarrow (n : \mathbb{N}) \rightarrow f\ n \sqsubseteq d$

$\text{antisym} : (d\ d' : A_{\perp}) \rightarrow d \sqsubseteq d' \rightarrow d' \sqsubseteq d \rightarrow d = d'$

$\text{trunc} \quad : (\xi\ \zeta : d \sqsubseteq d') \rightarrow \xi = \zeta$

# Algebraic syntax for a programming language

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

# 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

$$\begin{array}{c} \overline{\vdash}. \quad \frac{\Gamma \vdash A}{\vdash \Gamma, x : A} \quad \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma}{\Gamma \vdash U} \quad \frac{\Gamma \vdash a : U}{\Gamma \vdash \underline{a}} \\[1em] \frac{\Gamma \vdash a : U \quad \Gamma, x : \underline{a} \vdash B}{\Gamma \vdash (x : a) \Rightarrow B} \quad \frac{\Gamma \vdash t : (x : a) \Rightarrow B \quad \Gamma \vdash u : \underline{a}}{\Gamma \vdash t @ u : B[x \mapsto u]} \\[1em] \frac{\Gamma \vdash u : \underline{a} \quad \Gamma \vdash v : \underline{a}}{\Gamma \vdash u = v} \quad \dots \end{array}$$

# A domain-specific language for QIT signatures

$$\begin{array}{c} \overline{\vdash}. \quad \frac{\Gamma \vdash A}{\vdash \Gamma, x : A} \quad \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \quad \frac{\vdash \Gamma}{\Gamma \vdash U} \quad \frac{\Gamma \vdash a : U}{\Gamma \vdash \underline{a}} \\[2ex] \frac{\Gamma \vdash a : U \quad \Gamma, x : \underline{a} \vdash B}{\Gamma \vdash (x : a) \Rightarrow B} \quad \frac{\Gamma \vdash t : (x : a) \Rightarrow B \quad \Gamma \vdash u : \underline{a}}{\Gamma \vdash t @ u : B[x \mapsto u]} \\[2ex] \frac{\Gamma \vdash u : \underline{a} \quad \Gamma \vdash v : \underline{a}}{\Gamma \vdash u = v} \quad \dots \end{array}$$

A signature is a context  $\Gamma$ , e.g.

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

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



# This is a QIT itself

Con : Type  
Ty : Con  $\rightarrow$  Type  
Var : Con  $\rightarrow$  Type  
Tm : ( $\Gamma$  : Con)  $\rightarrow$  Ty  $\Gamma \rightarrow$  Type  
.  
: Con  
( $-$ ,  $-$  :  $-$ ) : ( $\Gamma$  : Con)  $\rightarrow$  Var  $\Gamma \rightarrow$  Ty  $\Gamma \rightarrow$  Con  
U : Ty  $\Gamma$   
= : Tm  $\Gamma$  U  $\rightarrow$  Ty  $\Gamma$   
( $-$  :  $-$ )  $\Rightarrow$  - : Var  $\Gamma \rightarrow$  ( $a$  : Tm  $\Gamma$  U)  $\rightarrow$  Ty ( $\Gamma$ ,  $x$  :  $\underline{a}$ )  $\rightarrow$  Ty  $\Gamma$   
- @ - : Tm  $\Gamma$  (( $x$  :  $a$ )  $\Rightarrow$  B)  $\rightarrow$  ( $u$  : Tm  $\Gamma$   $\underline{a}$ )  $\rightarrow$   
Tm  $\Gamma$  (B[x  $\mapsto$  u])  
...

# Results

- ▶ A generic definition of signatures for QITs which includes all the known examples
- ▶ Description of the induction principle
- ▶ 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



European Union

# THANK YOU FOR YOUR ATTENTION!

**SZÉCHENYI** 2020



HUNGARIAN  
GOVERNMENT

European Union  
European Social  
Fund



**INVESTING IN YOUR FUTURE**