An Order-Theoretic Analysis of Universe Polymorphism

Favonia (me), Carlo Angiuli, Reed Mullanix

Id Function

$\mathsf{id} : \forall a\,.\, a \longrightarrow a$

Id Function

$id : \forall a . a \rightarrow a$

with U, the universe, the type of types

Id Function in Dependent Type Theory

$id : (A{:}U) \rightarrow A \rightarrow A$

any type

Id Function

$id : \forall a . a \rightarrow a$

with U, the universe, the type of types

Id Function in Dependent Type Theory

$id : (A{:}U) \rightarrow A \rightarrow A$ id(U) can't work because U can't be in U

any type

Id Function

$id : \forall a \,.\, a \longrightarrow a$

with U, the universe, the type of types

Id Function in Dependent Type Theory

$id : (A{:}U) \longrightarrow A \longrightarrow A$ id(U) can't work because U can't be in U

any type

$id^{+} : (A{:}U^{+}) \longrightarrow A \longrightarrow A$ $id^{+}(U^{+})$ can't work because U^{+} can't be in U^{+}

any type...

including U

Agda, Lean

$$\mathrm{id} : (l : \mathrm{Level}) \longrightarrow (A : U_l) \longrightarrow A \longrightarrow A$$

any level

any type at level l

Agda, Lean

$id : (l : Level) \longrightarrow (A{:}U_l) \longrightarrow A \longrightarrow A$

any level — any type at level l

Coq, LEGO, Idris 1

$id : (A{:}U_?) \longrightarrow A \longrightarrow A$

system figuring levels out

Agda, Lean

$id : (l : Level) \longrightarrow (A:U_l) \longrightarrow A \longrightarrow A$

any level

any type at level l

Coq, LEGO, Idris 1

$id : (A:U_?) \longrightarrow A \longrightarrow A$

system figuring levels out

Matita

universe constraint $U_a < U_b$

$id : (A:U_b) \longrightarrow A \longrightarrow A$

Crude but Effective Stratification

by Conor McBride

So trivial to implement
Works well in practice

Theorem: This is also the most general*****

Crude but Effective Stratification

by Conor McBride

$$id : (A : U_0) \to A \to A$$

$$id^{\Uparrow n} : (A : U_n) \to A \to A$$

$id^{\Uparrow 1}(U_0)$ works!

Crude but Effective Stratification

by Conor McBride

$$id : (A : U_0) \to A \to A$$

$$id^{\Uparrow n} : (A : U_n) \to A \to A$$

$id^{\Uparrow 1}(U_0)$ works!

This design is also the most general*****

Universes in Type Theory

$A : U$ <=> A is a type

If $A : U$ and $B : U$, then $A \times B : U$, $A + B : U$, …

$U_0 : U_1 : U_2$ …

If $A : U_i$ then $A : U_{i+1}$

Universes in Cool Type Theory

$A : U$ <=> A is a type

If $A : U$ and $B : U$, then $A \times B : U$, $A + B : U$, …

~~$U_0 : U_1 : U_2$ …~~ $U_i : U_j$ whenever $i < j$

~~If $A : U_i$ then $A : U_{i+1}$~~ If $A : U_i$ then $A : U_j$ whenever $i <= j$

Levels can be any partially-ordered set

Cool Universe Levels

Natural numbers (boring)

Integers

… < -2 < -1 < 0 < 1 < 2 < …

id(U_{-1}) already worked without polymorphism

Rational numbers

… < 0 < … < 1 < …

"$(A : U_a) \to (B : U_b) \to$ … whenever $a < b$" is equivalent to "$(A : U_0) \to (B : U_1) \to$ …"

L-type theory

well-typed terms e

↓

L*-type theory

well-typed terms e*

L-type theory

well-typed terms e

↓ preserving <

L*-type theory

well-typed terms e*

L-type theory

well-typed terms e

↓ preserving <

L*-type theory

well-typed terms e*

category

StrictOrder

posets with
<-preserving maps

Universe Polymorphism

Levels with variables

= Monads on StrictOrder

posets with
<-preserving maps

Universe Polymorphism

Levels with variables

= Monads on StrictOrder

Theorem: You can embed any monad
on StrictOrder into McBride's scheme***

Crude but Effective Stratification

As a Monad

$$id^{a} : (A : U_{a}) \to A \to A$$

Crude but Effective Stratification

As a Monad

$$id^{a} : (A : U_{a}) \to A \to A$$

$$id^{a+n} : (A : U_{a+n}) \to A \to A$$

Crude but Effective Stratification

As a Monad

$$id^{a} : (A:U_{a}) \to A \to A$$

$$id^{a+n} : (A:U_{a+n}) \to A \to A$$

every level can be represented by (a,n)

The McBride Monad

$$\left[\begin{aligned} M(\Delta) &= \Delta \text{ “}\times\text{” } \mathbb{N} \\ \mathrm{return}(a) &= (a, 0) \\ \mathrm{join}((a,n_1),n_2) &= (a, n_1 + n_2) \end{aligned}\right]$$

every level can be represented by (a,n)

The Generalized McBride Monad

$M(\Delta) = \Delta$ “×” D

$\mathrm{return}(a) = (a, ✪)$

$\mathrm{join}((a, n_1), n_2) = (a, n_1 \bullet n_2)$

Works for any monoid (D, ✪, •) with a partial order $<$ such that $x < y$ implies $z \bullet x < z \bullet y$

Universality Theorem

You can embed* any monad on StrictOrder
into the (generalized) McBride monad
by choosing good (D, <, ✪, •)

Crude but Effective and Universal

Reusable OCaml Library

github.com/RedPRL/mugen

with many cool (D, <, ✪, •) like “fractals”

Partial Agda Mechanization

github.com/RedPRL/agda-mugen

“無限 mugen” means “infinity” in Japanese

Reusable OCaml Library

github.com/RedPRL/mugen

with many cool (D, <, ✪, •) like “fractals”

Demo: algaett

github.com/RedPRL/algaett

“algae” for algebraic effects