Université de Montréal

The case for Impredicative Universe Polymorphism

Stefan Monnier

monnier@iro.umontreal.ca

Université de Montréal

DIRO

Impredicativity 101

According to the O.E.D:

> im- + predicative, adj. & n.: With a sneaky form of circularity
>
> ∼1389, Chaucer: *Can't trust this dude, he's too impredicative*!

The origin of the notion of *types*, from Russell:

> Let S be the set of all sets that do not contain themselves:
>
> Does S contain itself?

Fix: introduce a stratification to prevent such self-applications

Anti-fix: some forms of impredicativity seem consistent and useful

Why do I care?

Working on Typer, an ML/Haskell with dependent types and macros

Typer: low-level λ-calculus intermediate language

Impredicativity used in:

- Encoding of modules into tuples
  (containing level-polymorphic definitions)
- Closure conversion
- The desire to subsume System F

Existing forms of impredicativity don't seem sufficient

Not fond of a special Prop universe (and didn't know about PR)

Forms of impredicativity

Impredicative universes: $\tau_2 : \text{Prop} \implies (x:\tau_1) \to \tau_2 : \text{Prop}$

As present in System F, Coq, Lean, and many others.

Resizing axioms: $\tau : \text{Type}_u \wedge P(\tau) \implies \tau : \text{Type}_{u'}$

Most famously, HoTT's propositional resizing.

Unsound: $\text{Type} : \text{Type}$

Clearly not ideal, especially with erasure.

New, IUP:

$$\frac{\Gamma, l : \text{Level} \vdash \tau : \text{Type}_u}{\Gamma \vdash (l : \text{Level}) \to \tau : \text{Type}_{u[0/l]}}$$

The present suggestion

Plan

Voices in my head:

- Encoding inductive types as closures.
- Encoding closures as inductive types.

Bounds:

- Strong sums defeat stratification.
- Encode System F

Encouraging signs

- Girard's Paradox did not bite (yet?).

Université de Montréal

Encoding inductive types as closures

Church-style encoding of lists:

$$\textit{List}\,\tau \quad = \quad (t : \mathsf{Type}) \rightarrow t \rightarrow (\tau \rightarrow t \rightarrow t) \rightarrow t$$

- No induction principle, hence no reasoning.

  Solutions by Awodey et.al. [2018] and Firsov and Stump [2018].

- No strong elimination.

  Limited solution by Jenkins et.al. [2021]

Strong elimination via universe polymorphism:

$$\textit{List}\,\tau \quad = \quad (l : \mathsf{Level}) \rightarrow (t : \mathsf{Type}_l) \rightarrow t \rightarrow (\tau \rightarrow t \rightarrow t) \rightarrow t$$

Université de Montréal

Universe of encoded inductive types

$$(t : \mathsf{Type}_l) \rightarrow t \rightarrow (\tau \rightarrow t \rightarrow t) \rightarrow t \quad : \quad \mathsf{Type}_{u \sqcup S\, l}$$

$$(l : \mathsf{Level}) \rightarrow (t : \mathsf{Type}_l) \rightarrow t \rightarrow (\tau \rightarrow t \rightarrow t) \rightarrow t \quad : \quad \mathsf{Type}_{??}$$

Predicative principles stipulate $\mathsf{sup}_l\ (u \sqcup S\ l)$:

$$(l : \mathsf{Level}) \rightarrow (t : \mathsf{Type}_l) \rightarrow \ldots \quad : \quad \mathsf{Type}_\omega$$

Yet! The type is isomorphic to the inductive: *List* $\tau \quad : \quad \mathsf{Type}_u$

IUP uses $\mathsf{inf}_l\ (u \sqcup S\ l)$:

$$(l : \mathsf{Level}) \rightarrow (t : \mathsf{Type}_l) \rightarrow \ldots \quad : \quad \mathsf{Type}_{(u \sqcup 1)}$$

Encoding closures as inductive types

Closure conversion turns (open) functions into pairs of:

captured environment $\times$ closed function

$\lambda n.n+1 : \textsf{Int} \rightarrow \textsf{Int}$	$\lambda n.n + \textit{length Float temps} : \textsf{Int} \rightarrow \textsf{Int}$
$\Longrightarrow$	$\Longrightarrow$
$((), \lambda(\textit{env}, n).n+1)$	$((\textit{length}, \textit{Float}, \textit{temps}),$ $\lambda(\textit{env}, n).n + \textit{env}.1\ \textit{env}.2\ \textit{env}.3)$

Hide the type of *env* to preserve types:

$$\textsf{Int} \rightarrow \textsf{Int} \Longrightarrow \exists t.(t \times ((t \times \textsf{Int}) \rightarrow \textsf{Int}))$$

Works great in System F! (after erasing *Float*)

Closure conversion with universes

With universes this turns into: $\exists(t : \mathsf{Type}_u).(t \times ((t \times \mathsf{Int}) \to \mathsf{Int}))$

And we need to hide u which depends on the captured environment:

$$\exists(l : \mathsf{Level}).\exists(t : \mathsf{Type}_l).(t \times ((t \times \mathsf{Int}) \to \mathsf{Int}))$$

Predicative principles stipulate $\mathsf{sup}_l\ ((S\ l)\ \sqcup\ 0)$:

$$\exists(l : \mathsf{Level}).\exists(t : \mathsf{Type}_l).\ldots \quad : \quad \mathsf{Type}_\omega$$

Yet! The type is equivalent to the arrow type: $\mathsf{Int} \to \mathsf{Int} \quad : \quad \mathsf{Type}_0$

IUP uses $\mathsf{inf}_l\ ((S\ l)\ \sqcup\ 0)$:

$$\exists(l : \mathsf{Level}).\exists(t : \mathsf{Type}_l).\ldots \quad : \quad \mathsf{Type}_1$$

Strong sums

Let's try IUP with strong sums:

$$\frac{\Gamma, l : \mathsf{Level} \vdash \tau : \mathsf{Type}_u}{\Gamma \vdash \Sigma l.\tau : \mathsf{Type}_{u[0/l]}}$$

We can define:

$$\textit{lower}\ (l : \mathsf{Level})\ (t : \mathsf{Type}_l)\ (x : t) = \langle l, \langle t, x \rangle \rangle$$

$$\textit{raise}\ (b : \Sigma l.\Sigma t : \mathsf{Type}_l.t) = b.2.2$$

This gives us:

$$\textit{lower}\ u\ \tau\ x \quad : \quad \Sigma l.\Sigma t : \mathsf{Type}_l.t \quad : \quad \mathsf{Type}_1$$

$$\forall x : \tau : \mathsf{Type}_u. \quad \textit{raise}\ (\textit{lower}\ u\ \tau\ x) \leadsto x$$

We can smuggle any value in a box that lives in Type_1!

Suggests that IUP is incompatible with first-class universe levels.

Université de Montréal

System F

IUP is as strong as System F:

$$
\begin{aligned}
[\![\forall t.\tau]\!] &= (l : \mathsf{Level}) \rightarrow (t : \mathsf{Type}_l) \rightarrow [\![\tau]\!] \\
[\![\Lambda t.e]\!] &= \lambda(l : \mathsf{Level}).\lambda(t : \mathsf{Type}_l).[\![e]\!] \\
[\![e[\tau]]\!] &= [\![e]\!]\ u\ [\![\tau]\!]
\end{aligned}
$$

We can just compute u from $[\![\tau]\!]$

DIRO

Well ordering

Common example of inconsistency in impredicative systems:

$$\begin{aligned}\textit{Ordering} : \text{Type} = {} & \Sigma(\textit{set} : \text{Type}_u).\\ & \Sigma(\textit{less-than} : \textit{set} \to \textit{set} \to \text{Type}).\\ & \ldots\end{aligned}$$

The inconsistency appears when we define an ordering of orderings.

In a predicative setting this does not work because *Ordering* ends up in a universe level higher than u.

If we want to try and reproduce the paradox using IUP, we need to abstract over the universe level of *set*.

Well ordering via existential quantification

First attempt:

$$\begin{aligned}\textit{Ordering1} : \mathsf{Type}_1 = \exists(l : \mathsf{Level}).\ & \Sigma(\textit{set} : \mathsf{Type}_l). \\ & \Sigma(\textit{less-than} : \textit{set} \to \textit{set} \to \mathsf{Type}_0). \\ & \ldots\end{aligned}$$

We *can* now instantiate *set* to this type.

But the weak nature of the existential makes *Ordering1* unusable:

We cannot eliminate to anything that depends on l, so ...

We cannot eliminate to anything that depends on *set*, so ...

Well ordering via universal quantification

Second attempt:

$$\begin{aligned} \mathit{Ordering2} : \mathsf{Type}_1 \;=\; & (l : \mathsf{Level}) \to \\ & \Sigma(\mathit{set} : \mathsf{Type}_l). \\ & \Sigma(\mathit{less\text{-}than} : \mathit{set} \to \mathit{set} \to \mathsf{Type}_0). \\ & \ldots \end{aligned}$$

Again, we can now instantiate *set* to this type (when l is 1).

¿Write a function which instantiates *set* to *Ordering2* when l is 1 yet to something in Type_0 when l is 0?

Use $\mathit{set} : \mathsf{Type}_{S\ l}$ to avoid the Type_0 case? Pushes *Ordering2* to Type_2!

Conclusion

Church-encoding suggests: $(l : \mathsf{Level}) \to \tau : \mathsf{Type}_{u[0/l]}$

Closure conversion suggests: $\exists(l : \mathsf{Level}).\tau : \mathsf{Type}_{u[0/l]}$

Better stop before $\Sigma(l : \mathsf{Level}).\tau : \mathsf{Type}_{u[0/l]}$!

IUP is as strong as System F.

We have failed to encode known paradoxes so far.

We have used only $(l : \mathsf{Level}) \to (t : \mathsf{Type}_l) \to \ldots$ so far

¡Help!