# Towards a Syntactic Model of Sized Dependent Types

Jonathan Chan (he/him)
University of British Columbia

# Why Dependent Types?

| | | |
|---|---|---|
| (Dependent) type theory | | (Predicate) logic |
| Types | correspond to | Propositions |
| Terms | | Proofs |
| Type checking | | Automated proof checking |

```
fix minus (n m: nat): nat ≔
  case n, m of
  | zero, _ ⇒ n
  | _, zero ⇒ n
  | succ p, succ q ⇒
    minus p q
```

```
fix div (n m: nat): nat ≔
  case n of
  | zero ⇒ zero
  | succ k ⇒
    succ (div (minus k m) m)
```

not structurally smaller than n!

Things that don't pass checks but should

```
fix minus (n m: nat): nat ≔
  case n, m of
  | zero, _ ⇒ zero
  | _, zero ⇒ n
  | succ p, succ q ⇒
    minus p q
```

```
fix div (n m: nat): nat ≔
  case n of
  | zero ⇒ zero
  | succ k ⇒
    succ (div (minus k m) m)
```

*still* not structurally smaller than n!

Things that don’t pass checks <u>even with inlining</u> but should

# Sized Types

additional size information

```
data nat [s] ≔
| zero: ∀ α < s. nat [s]
| succ: ∀ α < s. nat [α] → nat [s]
```

larger size

# Sized Types

```
data nat [s] ≔
| zero: ∀ α < s. nat [s]
| succ: ∀ α < s. nat [α] → nat [s]

fix minus [r] [s] (n: nat [r]) (m: nat [s]): nat [r] ≔ …

fix div [r] [s] (n: nat [r]) (m: nat [s]): nat [r] ≔
  case n of
  | zero [α] ⇒ zero [α]
  | succ [α] k ⇒ succ (div [α] [s] (minus [α] [s] k m) m)
```

recursion on smaller size α < r

# Past Work on Sized Types

$(\forall\ \alpha.\ \tau) \to \tau$ $\quad$ $\forall\ \alpha < s.\ \tau$

| Past work | Based on | Higher-rank sizes | Bounded sizes |
|---|---|---|---|
| Barthe et al. (2006), Grégoire et al. (2010), Sacchini (2011, 2013) | ✅ CIC | ❌ | ❌ |
| Abel (2006, 2012), Abel and Pientka (2016) | ❌ System Fω | ✅ | ✅ |
| Abel et al. (2017) | ✅ MLTT | ✅ | ❌ |
| This! Sized CCω (2021) | ✅ CCω | ✅ | ✅ |

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\begin{array}{l}\text{Sized CC}\omega \\ = \text{CC}\omega \\ + \text{higher-rank, bounded sizes} \\ + \text{naturals} + \text{W types}\end{array} \arrow[d, "\begin{array}{l}\Gamma \vdash e : \tau \\ \text{type-preservingly} \\ \text{compiles to} \\ [\![\Gamma]\!] \vdash [\![e]\!] : [\![\tau]\!]\end{array}"] & \text{consistency} \arrow[l, "\text{of}"'] \\
\text{Extensional CIC} \arrow[r, "\text{is known to be}"'] & \text{consistent} \arrow[u, "\begin{array}{c}\cdot \nvdash e : \bot \\ \text{implies} \\ \cdot \nvdash [\![e]\!] : \bot\end{array}"']
\end{tikzcd}$$

$$
\frac{\Phi,\ \alpha;\ \Gamma \vdash \sigma : \mathsf{Type} \qquad \Phi,\ \alpha;\ \Gamma,\ f\!:\ \overbrace{\forall\ \beta < \alpha.\ \tau[\alpha \mapsto \beta]}^{\text{IH}} \vdash e : \tau}{\Phi;\ \Gamma \vdash \mathsf{fix}\ f\ \underbrace{[\alpha]\!:\ \tau}_{\text{motive}} \coloneqq e : \forall\ \underbrace{\alpha.\ \tau}_{\text{QED}}}
$$

$$
\begin{aligned}
\mathsf{wfind} : {} & (P\!:\ \overbrace{\mathsf{Size} \to \mathsf{Type}}) \to \\
& ((\alpha\!:\ \mathsf{Size}) \to \overbrace{((\beta\!:\ \mathsf{Size}) \to \beta < \alpha \to P\ \beta)}^{\text{IH}} \to P\ \alpha) \to \\
\text{QED}\ \{\ & (\alpha\!:\ \mathsf{Size}) \to P\ \alpha \coloneqq \ldots
\end{aligned}
$$

## Challenges and Future Work

- Universe levels of `Size`s don't line up with those of translated inductives
- The infinite size is not well-founded and has no good translation
- Extend to general inductives/coinductives

## Questions?

Drop by the SRC virtual discussion session to ask them!

## Thank you!