A Syntactic Model for Sized Dependent Types

A Syntactic Model for Sized Dependent Types

a type-based method for
termination checking
recursive functions on inductive data

Sized TT

type-preserving translation

$\Gamma \vdash e : \tau$

$\Rightarrow [\![[\![\Gamma]\!]]\!] \vdash [\![[\![e]\!]]\!] : [\![[\![\tau]\!]]\!]$

extensional CIC

+ eta laws

Sized TT

- consistency
- normalization

type-preserving translation

$\Gamma \vdash e : \tau$

$\Rightarrow [\![\Gamma]\!] \vdash [\![e]\!] : [\![\tau]\!]$

extensional CIC + eta laws has

- consistency $(\cdot \nvdash e : \bot)$
- normalization $(\exists e'.\ \Gamma \vdash e \rhd^* e' \not\rhd^*)$

Outline

1. Sized Types - what & why
2. Features of the Sized TT
3. Key Translations
4. Remaining Work

1. Sized Types

data Nat [s] : Set where

  zero : $\forall r < s.$ Nat [s]

  succ : $\forall r < s.$ Nat [r] $\to$ Nat [s]

data W [s] (A : Type) (B : A $\to$ Type) : Type where

  sup : $\forall r < s.$ (a : A) $\to$ (B a $\to$ W [r] A B)

  $\to$ W [s] A B

1. Sized Types

base size

\circ

$$\text{one} \triangleq \text{succ}\,[\circ+1]\,(\text{zero}\,[\circ]) \quad : \text{Nat}\,[\circ+2]$$

$$\text{two} \triangleq \text{succ}\,[\circ+2]\ \text{one} \quad : \text{Nat}\,[\circ+3]$$

$$: \text{Nat}\,[\circ+4]$$

Since $\circ+2 < \circ+3 < \circ+4$

$(\text{div } n\ m \simeq \lceil \frac{n}{m+1} \rceil)$

1. Sized Types

fix minus (n : Nat) (m : Nat) : Nat .

fix div (n : Nat) (m : Nat) : Nat :=

match n with

| (zero ⇒ zero)

| (succ k ⇒

succ (div $\overbrace{(\text{minus } k\ m)}^{k-m\ \leq\ k}$ m)

not a syntactically smaller rec. call

1. Sized Types

$(\mathrm{div}\ n\ m \simeq \lceil \frac{n}{m+1} \rceil)$

fix minus [r] [s] (n : Nat [r]) (m : Nat [s]) : Nat [r].

fix div [r] [s] (n : Nat [r]) (m : Nat [s]) : Nat [r] :=

match n with

(zero [α] ⇒ zero [α])

(succ [α] k ⇒ $\alpha < r$

$$\mathrm{succ}\ [\alpha]\ (\mathrm{div}\ [\alpha]\ [s]\ (\underbrace{\mathrm{minus}\ [\alpha]\ [s]\ k\ m}_{\mathrm{Nat}\ [\alpha]})\ m)$$

2. Features

size vars

$$r, s ::= \alpha \mid \circ \mid \hat{s} \qquad s+1 \equiv \hat{s}$$

dep. funs, dep. pairs, let exps.

$$\sigma, \tau, e ::= \ldots \mid \forall \alpha . \tau \mid \forall \alpha < s . \tau$$

$$\mid \Lambda \alpha . e \mid \Lambda \alpha < s . e \mid e\,[s]$$

$$\mid \mathbb{N}\,[s] \mid \mathrm{zero}_{\mathbb{N}[s]}\,[r] \mid \mathrm{succ}_{\mathbb{N}[s]}\,[r]\;e$$

$$\mid \mathcal{W} x : \sigma . \tau\,[s] \mid \mathrm{sup}_{\mathcal{W} x : \sigma . \tau\,[s]}\,[r]\;e\;e$$

$$\mid \mathrm{match}\;e\;\mathrm{return}\;\lambda x . P\;\mathrm{with}\;(c\;z \ldots \Rightarrow e) \ldots$$

$$\mid \mathrm{fix}\;f\;[\alpha] : \sigma := e$$

$$\Gamma ::= \cdot \mid \Gamma(x : \tau) \mid \Gamma(x := e) \mid \qquad \Phi ::= \cdot \mid \Phi(\alpha) \mid \Phi(\alpha < s)$$

2. Features

size vars

$r, s ::= \alpha \mid \circ \mid \hat{s}$ $\quad\quad$ $s+1 \equiv \hat{s}$

$\sigma, \tau, e ::= \ldots$ | (bounded) size quantification/abstraction,

dep. funs, dep. pairs, let exps.

| naturals, W types,

| match expressions,

| fixpoints

| $\text{fix } f\, [\alpha] : \sigma := e$

$\Gamma ::= \cdot \mid \Gamma(x : \tau) \mid \Gamma(x := e) \mid$ $\quad\quad$ $\Phi ::= \cdot \mid \Phi(\alpha) \mid \Phi(\alpha < s)$

$$\dfrac{\phi(\alpha);\Gamma \vdash \sigma : \mathrm{Type} \qquad \phi(\alpha);\Gamma\,(f : \forall \beta < \alpha.\ \sigma[\alpha \mapsto \beta]) \vdash e : \sigma}{\phi;\Gamma \vdash \mathrm{fix}\ f\ [\alpha] : \sigma := e \ :\ \forall \alpha.\ \sigma}$$

$\forall \beta . (\forall \alpha . \sigma) \to \tau$ $\forall \alpha < s . \tau$

2. Features

	HO Sizes	Bounded	Dependent	Coinds.
Sacchini	✗ implicit	✗	✓	✓
Abel (2017) NbE for SDT	✓	✗	✓	✗ Nats
Abel (2016) Wf Rec.	✓	✓	✗ F_ω	✓ μ, ν
Agda	✓	✓	✓	✓
this	✓	✓	✓	✗ N, W

$\forall \beta.\, (\forall \alpha.\, \sigma) \to \tau \qquad \forall \alpha < s.\, \tau$

2. Features

	HO Sizes	Bounded	∞ Size	Consistent?
Sacchini	✗ implicit	✗	✓	✓
Abel (2017) NbE for SDT	✓	✗	✓	✓
Abel (2016) Wf Rec.	✓	✓	✓	✓
Agda	✓	✓	✓	✗ :(
this	✓	✓	✗	✓?

$$\mathrm{data}\ \ \mathrm{Nat}\,[s] : \mathrm{Set}\ \ \mathrm{where}$$
$$\quad \mathrm{zero} : \forall r < s.\ \mathrm{Nat}\,[s]$$
$$\quad \mathrm{succ} : \forall r < s.\ \mathrm{Nat}\,[r] \to \mathrm{Nat}\,[s]$$

In eCIC :

- Size is an inductive type
- $\_ \leqslant \_$ is also an (indexed) inductive type
- Nat is parametrized by a Size s
- zero, succ take a Size r and a proof $r < s$
  $\triangleq r+1 \leqslant s$

3. Translation

$$
\begin{aligned}
&\mathsf{data}\ \mathrm{Nat}\,[s] : \mathrm{Set}\ \ \mathsf{where} \\
&\quad \mathrm{zero} : \forall r < s.\ \mathrm{Nat}\,[s] \\
&\quad \mathrm{succ} : \forall r < s.\ \mathrm{Nat}\,[r] \to \mathrm{Nat}\,[s]
\end{aligned}
$$

Suppose: $n : (\alpha : \mathrm{Size}) \times \mathrm{Nat}\ \alpha$

$$
\mathrm{Succ}\ \underbrace{(\mathrm{fst}\ n + 1)}_{s}\ \underbrace{(\mathrm{fst}\ n)}_{r}\ \_\ (\mathrm{snd}\ n) : \mathrm{Nat}\ \overbrace{(\mathrm{fst}\ n + 1)}^{s}
$$

$$
\begin{gathered}
\mathrm{fst}\ n < \mathrm{fst}\ n + 1 \\
r < s
\end{gathered}
$$

data W [s] (A : Type) (B : A → Type) : Type where

sup : ∀r < s. (a : A) → (B a → W [r] A B) → W [s] A B

Suppose: $a : A$, $f : B\,a \to (\alpha : \mathrm{Size}) \times W \alpha A B$

$\mathrm{sup}(?+1)\, A\, B\, ?\, \_\, a\, (\mathrm{snd} \circ f) : W\,(?+1)\, A\, B$

$\mathrm{fst} \circ f : B\,a \to \mathrm{Size}$

$? : \mathrm{Size}$

```
data Size : Type where
  ↑_ : Size → Size
  ⊔_ : {A : Type} → (A → Size) → Size
```

the "limit" or least upper bound of all the sizes returned by A → Size mono : $\alpha \le \beta \to {\uparrow}\alpha \le {\uparrow}\beta$

Monotonicity of $\uparrow$ wrt $\le$

cocone : $(a : A) \to \beta \le f\,a \to \beta \le \sqcup f$

$\sqcup f$ is an upper bound : $f\,a \le \sqcup f$

limit : $((a : A) \to f\,a \le \beta) \to \sqcup f \le \beta$

$\sqcup f$ is a <u>least</u> upper bound

Well-founded induction on Size:

(via accessibility predicate wrt strict order $<$)

$$\begin{aligned}
\mathrm{elim} : \; & (P : \mathrm{Size} \to \mathrm{Type}) \to \\
& ((\alpha : \mathrm{Size}) \to \underbrace{((\beta : \mathrm{Size}) \to \beta < \alpha \to P\,\beta)}_{\mathrm{IH}} \to P\,\alpha) \to \\
& (\alpha : \mathrm{Size}) \to P\,\alpha
\end{aligned}$$

$$\frac{\phi(\alpha);\Gamma \vdash \sigma : \mathrm{Type}^{\text{IH}} \qquad \phi(\alpha);\Gamma(f : \overbrace{\forall\beta<\alpha.\ \sigma[\alpha\mapsto\beta]}^{\text{IH}}) \vdash e : \sigma}{\phi;\Gamma \vdash \mathrm{fix}\ f\ \underbrace{[\alpha] : \sigma}\ := e \ : \ \forall\alpha.\ \sigma}$$

$$\mathrm{elim} : (\overbrace{P : \mathrm{Size} \to \mathrm{Type}}) \to$$

$$((\alpha:\mathrm{Size}) \to \underbrace{((\beta:\mathrm{Size}) \to \beta<\alpha \to P\beta)}_{\text{IH}} \to P\alpha) \to$$

$$(\alpha:\mathrm{Size}) \to P\alpha$$

$$[\![\text{fix } f\,[\alpha] : \sigma := e]\!] =$$

$$\text{elim } (\lambda \alpha : \text{Size}.\, [\![\sigma]\!])$$
$$(\lambda \alpha : \text{Size}.\ \lambda f : (\beta : \text{Size}) \to [\![\sigma[\alpha \mapsto \beta]]\!].\ [\![e]\!])$$
$$: (\alpha : \text{Size}) \to [\![\sigma]\!]$$

Technical details:

- Need equality reflection to show def. eq.

$$acc1, acc2 : \mathrm{Acc}\ \alpha \Rightarrow acc1 = acc2$$

- Need eta laws for match to show def. eq.

$$\begin{array}{ccc} [\![\mathrm{fix}\ f\ [\alpha] : \sigma := e]\!] & = & \mathrm{elim} \ldots \\ \rhd & & \| \\ [\![e[f \mapsto \mathrm{fix} \ldots]]\!] & = & e' \end{array}$$

Problem 1: ∞ Size

data Nat [s] : Set where ...

data Nat* : Type where ...

Nat [s] is an approximation of Nat*

$$\text{Nat}\,[\infty] = \text{Nat}^*$$

where for every s, $s < \infty \Rightarrow \infty < \infty$

$\therefore$ ∞ is not well-founded!

Option 1: Add ∞ anyway

- Avoid inconsistency by syntactically restricting when you're allowed to apply ∞ (Hughes 1996)
- Throw wf induction out :(
- Morally reprehensible

Option 2: Existential sizes + cast

- Use $\exists\alpha.\,\mathrm{Nat}[\alpha]$ in place of $\mathrm{Nat}[\infty]$
- Add axiom $\forall\alpha.\,\mathrm{Nat}[\alpha] \to \exists\alpha.\,\mathrm{Nat}[\alpha]$
  to recover property $\forall\alpha.\,\mathrm{Nat}[\alpha] < \mathrm{Nat}[\infty]$ $(\alpha < \infty)$
- ⚠ This is the axiom of choice — nonconstructive!
- Lose Canonicity
  - $\vdash e : \exists\alpha.\,\Upsilon \not\Rightarrow e \equiv \langle s, e' \rangle$

Option 3 : Size erasure

- Have both Nat and Nat* in source
  W and W*
- Add an operator that "erases" sizes
  e.g. erase div : Nat* → Nat*

  op. sem. are messy
- Match* can't be translated
- Unclear what erase translates to
- Erasure might be a modality?

Problem 2 : Size is too Big

data Size : $\mathrm{Type}_{\ell+1}$ where

  $\uparrow\_ \;:\; \mathrm{Size} \to \mathrm{Size}$

  $\sqcup\_ \;:\; \{A : \mathrm{Type}_{\ell}\} \to (A \to \mathrm{Size}) \to \mathrm{Size}$

data Nat (s : Size) : Type_{1} where ...

$(\ell = 0)$ ↳ should be Set!

(Type_0)

Option 1: Impredicative Set

data Size : Set where ...

$\sqcup\_ : \{A : \mathrm{Type}_\ell\} \to (A \to \mathrm{Size}) \to \mathrm{Size}$

- Cannot eliminate Size to Type
  $\Rightarrow$ cannot show that $\forall s, f,\ \uparrow s \neq \sqcup f$
- Morally reprehensible to classical mathematicians

Option 2: Larger Types

data Nat [α] : Type, where ...

- Morally reprehensible to me
  - ↳ exposes "impl." (proof) details

Option 3: Parametrized Size

data Size (A : $Type_\ell$) : $Type_\ell$ where ...

data Nat (s : Size $\perp$) : Set where ...

↳ $\ell = 0$

- Complicates translation & proofs
- What about arbitrary size quantifications? appls.?

$$[\![\forall\alpha.\tau]\!] = (s : \text{Size } ?) \to [\![\tau]\!]$$

$$[\![e\ [s]]\!] = [\![e]\!]\ [\![s]\!]$$

↳ : Size ?

Future work:

- Coinductives e.g. cofix inf [α] (a : A) : Stream [α] A :=
  {hd [$\beta < \alpha$] := a;
  tl [$\beta < \alpha$] := inf [β] a}

- (Co)inductives in general i.e. data X [α] ... : Υ where ...

- Some size inference to reduce annot. burden

- Decidability of type checking, confluence, SR

FIN