Type system for a stack-based programming language

February 25, 2025

Introduction

TBA

Language

Declarations

Data declaration

The data types are represented by Algebraic Data Types (ADT) in the language.

```
data Either a b:
  [a] left,
  [b] right.
```

If a constructor does not take any types its input parameters can be omitted.

```
data Maybe a:
  nothing.
  [a] just,
```

ADT's can be recursive.

```
data Nat:
  zero,
  [Nat] suc.
```

Operator declaration

The operators are represented by a sequence of operators. All operator definitions must have a type annotation.

```
define [Nat, Nat] add [Nat]:
  case { zero { }, suc { add suc } }.

define [Nat] addThree [Nat]:
  zero suc suc suc add.
```

An operator's body can be empty

```
define [] nop []:.
```

Prelude

Prelude operators are split into two groups: parametric and non-parametric. Parametric operators take additional number parameters in their name.

Non-parametric

- `pop`
  Delete the top element.

- `dup`
  Duplicate the top element.

- `quote`
  Create a first class function from the top element.

Parametric

- `br-n`
  Bury the top element to n-th position.

- `dg-n`
  Dig up the n-th element to the top.

- `comp-x-y-z-w`
  Compose two FCF into one FCF. Takes two input arguments.
  First (topmost) input argument is an FCF that takes z input arguments, returns w output arguments, and is executed second.
  Second input argument is an FCF that takes x input arguments, returns y output arguments, and is executed first.

- `exec-x-y`
  First class function execution.
  First (topmost) input argument. is an FCF that takes x input arguments, returns y output arguments.
  The rest x input arguments are FCF input arguments.
  The y output arguments are FCF output arguments.

Type system

Operator Type separation

- **Type -** represents the type of a value stored on the stack.

- **Operator Type -** represents the type of an element of an operator body.

Notation

- "[*pre*][*post*]" represents an operator type. Where *pre* and *post* represent the input and output parameters.

- An operator(s) between the stack descriptions is a shorthand, e.g., writing $[a, b]$`foo bar`$[c, d]$ is equivalent to `foo bar` : $[a, b][c, d]$

- The leftmost element in the stack type description is the most recently pushed, e.g., $[]$`foo bar baz`$[Baz, Bar, Foo]$.

Most General Unifier for lists

$$
\begin{aligned}
&\text{listmgu } [\,]\ [\,] && = \{\} \\
&\text{listmgu } [t1 : r1]\ [t2 : r2] && = \text{let } s := \text{mgu } t1\ t2 \\
& && \quad\ \text{in listmgu } (s r1)\ (s r2)
\end{aligned}
$$

Specialization Rule (Type)

$$\frac{t' = \{a \mapsto a'\}t}{t \sqsubseteq t'} \tag{Type spec}$$

Specialization Rule (Operator Type)

$$\frac{[\alpha'][\beta'] = \{a \mapsto a'\}[\alpha][\beta]}{[\alpha][\beta] \sqsubseteq [\alpha'][\beta']} \tag{Op spec}$$

Operator Augmentation

$$\frac{[\alpha][\beta] \sqsubseteq [\alpha'][\beta']}{[\alpha][\beta] \sqsubseteq [\alpha' \cdot \gamma][\beta' \cdot \gamma]} \tag{Op aug}$$

Name rule

$$\frac{[\alpha']\mathtt{op}[\beta'] \in \Gamma \quad [\alpha][\beta] = \mathrm{inst}([\alpha'][\beta'])}{\Gamma \vdash [\alpha]\mathtt{op}[\beta]} \tag{Name}$$

Chain rule

$$\frac{\begin{array}{c}\mathrm{listmgu}(\beta, \psi) \\ \Gamma \vdash [\alpha']\mathtt{x}[\beta'] \sqsubseteq [\alpha][\beta] \quad \Gamma \vdash [\psi']\mathtt{y}[\omega'] \sqsubseteq [\psi][\omega]\end{array}}{\Gamma \vdash [\alpha]\mathtt{x\ y}[\omega]} \tag{Chain}$$

Case rule

$$\frac{\Gamma \vdash []\mathtt{constr1}^{-1}\ \mathtt{body1}[] \quad \Gamma \vdash [][]}{\Gamma \vdash [\alpha]\mathtt{case}\{\mathtt{constr1}\{\mathtt{body1}\}, \ldots\}[\beta]} \tag{Chain}$$

Comp rule

$$\frac{\begin{array}{c}\mathrm{listmgu}(\beta, \psi) \\ \Gamma \vdash [\alpha'][\beta'] \sqsubseteq [\alpha][\beta] \quad \Gamma \vdash [\psi'][\omega'] \sqsubseteq [\psi][\omega] \\ ||\alpha'|| = m \quad ||\beta'|| = n \quad ||\psi'|| = x \quad ||\omega'|| = y\end{array}}{\Gamma \vdash [[\alpha'][\beta'], [\psi'][\omega']]\mathtt{comp\text{-}m\text{-}n\text{-}x\text{-}y}[[\alpha][\omega]]} \tag{Comp}$$

Exec rule

$$\frac{||\alpha'|| = m \quad ||\beta'|| = n}{\Gamma \vdash [[\alpha][\beta], \alpha]\mathtt{exec\text{-}m\text{-}n}[beta]} \tag{Exec}$$