A Module / Functor Language

Alex Knauth, Milo Turner

Overview

Plan: build a prototype as a *core language*, then extend it with a *module language*.

The core language is a generic language for creating definitions and manipulating values.

The module language is the language for organizing definitions into modules and abstracting over modules with functors.

Core Language

- For the prototype: As simple as possible for a proof of concept.

```
core-defn = (type id = core-type)
          | (val id : core-type = core-expr)

core-type = Int
          | (-> core-type ... core-type)
          | (∀ (id ...) core-type)

core-expr = integer
          | (core-expr core-expr ...)
          | (λ ({id : core-type} ...) core-expr)
          | (Λ (id ...) core-expr)
          | (let ([id core-expr] ...) core-expr)
```

Module Language: *mod*s and *sig*s

```
toplevel-binding = (define-signature id = signature-expr)
                 | (define-module id = module-expr)

    module-expr = id
                 | (mod core-def ...)
                 | (seal module-expr :> signature-expr)

  signature-expr = id
                 | (sig sig-component ...)

   sig-component = (type id) ; opaque type declaration
                 | (type id = core-type)
                 | (val id : core-type)
```

Module Language: *functor*s and *Pi* signatures

```
module-expr = id
            | (mod core-def ...)
            | (seal module-expr :> signature-expr)
            | (module-expr module-expr)
            | (λ ({id : signature-expr}) module-expr)
            | (let ([id module-expr]) module-expr)

signature-expr = id
               | (sig sig-component ...)
               | (Π ({id : signature-expr}) signature-expr)
               | (let ([id module-expr]) signature-expr)
```

Example:

```
(define-signature IMG =
  (sig
   (type Img)
   (val above : (-> Img Img Img))
   (val beside : (-> Img Img Img))
   (val place-img : (-> Img Int Int Img Img))
   (val circle : (-> Int Img))))

(define-module Pict = (mod …))
(define-module 2HtDP = (mod …))
(define-module SVG = (mod …))
```

```
(define-module Monalisa =
  (λ ({I : IMG})
    (mod
     (val bg : I.Img = …)
     (val lisa : I.Img = …)
     (val mona : I.Img =
       (place-img lisa 30 65 bg)))))

(define-module Monalisa/Pict = (Monalisa Pict))
(define-module Monalisa/2htdp = (Monalisa 2HtDP))
(define-module Monalisa/SVG = (Monalisa SVG))
```

GitHub Repository

https://github.com/macrotypefunctors/typed-module-lang