Core Language

```
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

```
          program = toplevel-binding
                    ...

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

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

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

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