

## 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 = id
           | 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 = signature-expr)
                  | (define-module id = module-expr)

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)

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