HOT Compilation Notes

Rahul Manne
rmanne@andrew.cmu.edu

Disclaimer/README

These are only reference notes, and by no means fully capture what is taught in class.

Notes for 170131 (on substitution) are extremely incoherent so I did not include them by default.

There may be errors, feel free to report them to me.

1 Compiler Structure

SML
$\xrightarrow{\text{elaborate}}$ IL-Module
$\xrightarrow{\text{phase-splitting}}$ IL-Direct
$\xrightarrow{\text{cps conversion}}$ IL-CPS
$\xrightarrow{\text{closure conversion}}$ IL-Closure
$\xrightarrow{\text{hoisting}}$ IL-Hoist
$\xrightarrow{\text{allocation}}$ IL-Alloc
$\xrightarrow{\text{code-generation}}$ C

2 Introduction to the $F\omega$ type system

2.1 Grammar

$$\begin{aligned} k &::= \text{Type} \mid k \to k \\ c &::= \alpha \mid c \to c \mid \forall \alpha : k \,.\, c \mid c\ c \\ e &::= x \mid \lambda x : c \,.\, e \mid e\ e \mid \Lambda \alpha : k \,.\, e \mid e[c] \end{aligned}$$

Kind k, Type Constructor c, and Term e.

Note: Type is often referred to with just "T" (by Crary), for simplicity.

2.2 Context for Judgements

$$\Gamma ::= \epsilon \mid \Gamma, x : \tau \mid \Gamma, \alpha : k \tag{1}$$

Note: For simplicity, whenever a new α appears in the context, we implicitly ensure that α is not already in Γ.

2.3 $\Gamma \vdash c : k$

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha : k} \qquad \frac{\Gamma \vdash \tau : T \qquad \Gamma \vdash \tau_2 : T}{\Gamma \vdash \tau_1 \to \tau_2 : T} \qquad \frac{\Gamma, \alpha : k \vdash \tau : T}{\Gamma \vdash \forall \alpha : k \,.\, \tau} \qquad \frac{\Gamma, \alpha : k \vdash c : k'}{\Gamma \vdash \lambda \alpha : k \,.\, c : k \to k'}$$

$$\frac{\Gamma \vdash c_1 : k \to k' \qquad \Gamma \vdash c_2 : k}{\Gamma \vdash c_1\ c_2 : k'}$$

2.4 $\Gamma \vdash e : \tau$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x : \tau \,.\, e : \tau \to \tau'} \qquad \frac{\Gamma \vdash e_1 : \tau \to \tau' \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ e_2 : \tau'} \qquad \frac{\Gamma, \alpha : k \vdash e : \tau}{\Gamma \vdash \Lambda \alpha : k \,.\, e : \forall \alpha : k \,.\, \tau}$$

$$\frac{\Gamma \vdash e : \forall \alpha : k \,.\, e \qquad \Gamma \vdash c : k}{\Gamma \vdash e[c] : [c \,/\, \alpha]e} \qquad \frac{\Gamma \vdash e : \tau \qquad \Gamma \vdash \tau \equiv \tau' : T}{\Gamma \vdash e : \tau'}$$

2.5 $\Gamma \vdash c \equiv c : k$

Definitional Equivalence.

$$\frac{\Gamma \vdash c : k}{\Gamma \vdash c \equiv c : k} \qquad \frac{\Gamma \vdash c \equiv c' : k}{\Gamma \vdash c' \equiv c : k} \qquad \frac{\Gamma \vdash c_1 \equiv c_2 : k \qquad \Gamma \vdash c_2 \equiv c_2 : k}{\Gamma \vdash c_1 \equiv c_3 : k}$$

The above are identity, reflexivity, and transitivity respectively.

The following are "compatibility" rules.

$$\frac{\Gamma \vdash c_1 \equiv c_1' : k \qquad \Gamma \vdash c_2 \equiv c_2' : k}{\Gamma \vdash c_1 c_2 \equiv c_1' c_2' : k} \qquad \frac{\Gamma, \alpha : k_1 \vdash c \equiv c' : k_2}{\Gamma \vdash \lambda\alpha : k_1 \,.\, c \equiv \lambda\alpha : k_1 \,.\, c' : k_1 \to k_2}$$

$$\frac{\Gamma \vdash \tau_1 \equiv \tau_1' : k \qquad \Gamma \vdash \tau_2 \equiv \tau_2' : k}{\Gamma \vdash \tau_1 \to \tau_2 \equiv \tau_1' \to \tau_2' : T} \qquad \frac{\Gamma, \alpha : k \vdash \tau \equiv \tau' : T}{\Gamma \vdash \forall\alpha : k \,.\, \tau \equiv \forall\alpha : k \,.\, \tau' : T}$$

congruence = compatible equivalence relation
The following are the rules for beta equivalence and extensionality:

$$\frac{\Gamma \vdash c_2 : k \qquad \Gamma, \alpha : k \vdash c_1 : k'}{\Gamma \vdash (\lambda\alpha : k \,.\, c_1)\ c_2 \equiv [c_2 \,/\, \alpha]c_1 : k'} \qquad \frac{\Gamma, \alpha : k_1 \vdash c\ \alpha \equiv c'\ \alpha : k_2 \qquad \Gamma \vdash c : k_1 \to k_2 \qquad \Gamma \vdash c' : k_1 \to k_2}{\Gamma \vdash c \equiv c' : k_1 \to k_2}$$

2.6 Extending $F\omega$

Note: This helps in the understanding of sml's module system

Grammar:

$$\begin{aligned} k &::= \ldots \mid k \times k \\ c &::= \ldots \mid \langle c, c \rangle \mid \pi_1 c \mid \pi_2 c \end{aligned}$$

New Judgements:

$$\frac{\Gamma \vdash c_1 : k_2 \qquad \Gamma \vdash c_2 : k_2}{\Gamma \vdash \langle c_1, c_2 \rangle : k_1 \times k_2} \qquad \frac{\Gamma \vdash c : k_1 \times k_2}{\Gamma \vdash \pi_i\ c : k_1} \qquad \frac{\Gamma \vdash c_1 \equiv c_1' : k_1 \qquad \Gamma \vdash c_2 \equiv c_2' : k_2}{\Gamma \vdash \langle c_1, c_2 \rangle \equiv \langle c_1', c_2' \rangle : k_1 \times k_2} \qquad \frac{\Gamma \vdash c \equiv c' : k_1 \times k_2}{\Gamma \vdash \pi_i\ c \equiv \pi_i\ c' : k_i}$$

$$\frac{\Gamma \vdash c_1 : k_1 \qquad \Gamma \vdash c_2 : k_2}{\Gamma \vdash \pi_i \langle c_1, c_2 \rangle \equiv c_i : k_i} \qquad \frac{\Gamma \vdash \pi_1\ c \equiv \pi_1\ c' : k_1 \qquad \Gamma \vdash \pi_2\ c \equiv \pi_2\ c' : k_2}{\Gamma \vdash c \equiv c' : k_1 \times k_2}$$

3 Algorithmic Equivalence in the $F\omega$ Type System

3.1 Normalize-and-Compare

Note: We don't use this.

$\lambda\alpha : k \,.\, c_1 \; c_2 \xrightarrow{\beta} [c_2 \,/\, \alpha]c_1$
$\pi_i\langle c_1, c_2\rangle \xrightarrow{\beta} c_i$
$+$ some η reduction rules

According to some equivalence theorem, they will have normal forms and those normal forms will be equal if they are equivalent.

3.2 Grammar and Properties

Paths:
$p ::= \alpha \mid p \; c \mid \pi_1 \; p \mid \pi_2 \; p$

Weak-Head Normal Form:
$n ::= p \mid c_1 \to c_2 \mid \forall\alpha : k \,.\, c.$

Regularity:
If $\vdash \Gamma\,\text{ok}$ and $\Gamma \vdash c_1 \equiv c_2 : k$, then $\Gamma \vdash c_1 : k$ and $\Gamma \vdash c_2 : k$.
If $\vdash \Gamma\,\text{ok}$ and $\Gamma \vdash c : k$, then $\Gamma \vdash k : \text{kind}$.

Soundness:
If $\vdash \Gamma\,\text{ok}$ and $\Gamma \vdash c_1, c_2 : k$ and $\Gamma \vdash c_1 \Leftrightarrow c_2 : k$, then $\vdash c_1 \equiv c_2 : k$.

Completeness:
If $\vdash \Gamma\,\text{ok}$ and $\Gamma \vdash c_1 \equiv c_2 : k$, then $\Gamma \vdash c_1 \Leftrightarrow c_2 : k$.

$$\frac{}{\vdash \epsilon\,\text{ok}} \qquad \frac{\vdash \Gamma\,\text{ok} \qquad \Gamma \vdash k : \text{kind}}{\vdash \Gamma, \alpha : k\,\text{ok}} \qquad \frac{\vdash \Gamma\,\text{ok} \qquad \Gamma \vdash \tau : \text{Type}}{\vdash \Gamma, x : \tau\,\text{ok}}$$

3.3 Algorithmic Constructor Equivalence

Form: $\overset{+}{\Gamma} \vdash \overset{+}{c_1} \Leftrightarrow \overset{+}{c_2} : \overset{+}{k}$

Note: $\overset{+}{x}$ indicates that x is an input.

$$\frac{\Gamma, \alpha : k_1 \vdash c\ \alpha \Leftrightarrow c'\ \alpha : k_2}{\Gamma \vdash c \Leftrightarrow c' : k_1 \to k_2} \qquad \frac{\Gamma \vdash \pi_1\ c \leftrightarrow \pi_1\ c' : k_1 \qquad \Gamma \vdash \pi_2\ c \leftrightarrow \pi_2\ c' : k_2}{\Gamma \vdash c \Leftrightarrow c' : k_1 \times k_2}$$

$$\frac{c_1 \Downarrow c_1' \qquad c_2 \Downarrow c_2' \qquad \Gamma \vdash c_1' \leftrightarrow c_2' : \text{Type}}{\Gamma \vdash c_1 \Leftrightarrow c_2 : \text{Type}}$$

3.4 Algorithmic Path Equivalence

Form: $\overset{+}{\Gamma} \vdash \overset{+}{c_1} \Leftrightarrow \overset{+}{c_2} : \overset{-}{k}$

Note: $\overset{-}{x}$ indicates that x is an output.

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha \leftrightarrow \alpha : k} \qquad \frac{\Gamma \vdash p \leftrightarrow p' : k_1 \to k_2 \qquad \Gamma \vdash c \Leftrightarrow c' : k_1}{\Gamma \vdash p\ c \leftrightarrow p'\ c' : k_1} \qquad \frac{\Gamma \vdash p \leftrightarrow p' : k_1 \times k_2}{\Gamma \vdash \pi_i\ p \leftrightarrow \pi_i\ p' : k_i}$$

$$\frac{\Gamma \vdash c_1 \Leftrightarrow c_1' : T \qquad \Gamma \vdash c_1 \Leftrightarrow c_2' : T}{\Gamma \vdash c_1 \to c_2 \leftrightarrow c_1' \to c_2' : T} \qquad \frac{\Gamma, \alpha : k \vdash c \Leftrightarrow c' : T}{\Gamma \vdash \forall \alpha : k\ .\ c \leftrightarrow \forall \alpha : k\ .\ c' : T}$$

3.5 Weak-Head Normalization

Form: $\overset{+}{c} \Downarrow \overset{-}{n}$

$$\frac{c \rightsquigarrow c' \qquad c' \Downarrow c''}{c \Downarrow c''} \qquad \frac{c \not\rightsquigarrow}{c \Downarrow c}$$

3.6 Weak-Head Reduction

Form: $\overset{+}{c} \rightsquigarrow \overset{-}{c'}$

$$\frac{}{(\lambda \alpha : k\ .\ c_1)\ c_2 \rightsquigarrow [c_2\ /\ \alpha]c_1} \qquad \frac{}{\pi_i \langle c_1, c_2 \rangle \rightsquigarrow c_i} \qquad \frac{c_1 \rightsquigarrow c_1'}{c_1\ c_2 \rightsquigarrow c_1'\ c_2} \qquad \frac{c \rightsquigarrow c'}{\pi_i c \rightsquigarrow \pi_i c'}$$

3.7 Kind Synthesis and Checking

Form: $\overset{+}{\Gamma} \vdash \overset{+}{c} \Rightarrow \overset{-}{k}$ and $\overset{+}{\Gamma} \vdash \overset{+}{c} \Leftarrow \overset{+}{k}$

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha \Rightarrow k} \qquad \frac{\Gamma, \alpha : k \vdash c \Rightarrow k'}{\Gamma \vdash \lambda\alpha : k \,.\, c \Rightarrow k \rightarrow k'} \qquad \frac{\Gamma \vdash c_1 \Rightarrow k \rightarrow k' \qquad \Gamma \vdash c_2 \Leftarrow k}{\Gamma \vdash c_1\ c_2 \Rightarrow k'} \qquad \frac{\Gamma \vdash c_1 \Rightarrow k_1 \qquad \Gamma \vdash c_2 \Rightarrow k_2}{\Gamma \vdash \langle c_1, c_2 \rangle \Rightarrow k_1 \times k_2}$$

$$\frac{\Gamma \vdash c \Rightarrow k_1 \times k_2}{\Gamma \vdash \pi_i\ c \Rightarrow k_1} \qquad \frac{\Gamma \vdash c_1 \Leftarrow T \qquad \Gamma \vdash c_2 \Leftarrow T}{\Gamma \vdash c_1 \rightarrow c_2 \Rightarrow T} \qquad \frac{\Gamma, \alpha : k \vdash c \Leftarrow T}{\Gamma \vdash \forall\alpha : k \,.\, c \Rightarrow T} \qquad \frac{\Gamma \vdash c \Rightarrow k}{\Gamma \vdash c \Leftarrow k}$$

3.8 Type Checking and Synthesis

Form: $\overset{+}{\Gamma} \vdash \overset{+}{e} \Rightarrow \overset{-}{c}$ and $\overset{+}{\Gamma} \vdash \overset{+}{e} \Leftarrow \overset{+}{c}$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x \Rightarrow \tau} \qquad \frac{\Gamma \vdash \tau \Leftarrow T \qquad \Gamma, x : \tau \vdash e \Rightarrow \tau'}{\Gamma \vdash \lambda x : \tau \,.\, e \Rightarrow \tau \rightarrow \tau'} \qquad \frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \qquad \tau_1 \Downarrow \tau \rightarrow \tau' \qquad \Gamma \vdash e_2 \Leftarrow \tau}{\Gamma \vdash e_1\ e_2 \Rightarrow \tau'}$$

$$\frac{\Gamma, \alpha : k \vdash e \Rightarrow \tau}{\Gamma \vdash \Lambda\alpha :.\ e \Rightarrow \forall\alpha : k \,.\, \tau} \qquad \frac{\Gamma \vdash e \Rightarrow \tau \qquad \tau \Downarrow \forall\alpha : k \,.\, \tau' \qquad \Gamma \vdash c \Leftarrow k}{\Gamma \vdash e[c] \Rightarrow [c \,/\, \alpha]\tau'} \qquad \frac{\Gamma \vdash e \Rightarrow \tau' \qquad \Gamma \vdash \tau \Leftrightarrow \tau' : T}{\Gamma \vdash e \Leftarrow \tau}$$

4 Singleton Kinds

```
sig
  type t
  type 'a u
  type ('a,'b) v
  type w = int
  type w' = w
  .
  .
  .
end
```

To represent this in type our type system, $t : \text{Type}$
$u : \text{Type} \to \text{Type}$
$v : \text{Type} \to \text{Type} \to \text{Type}$
(or $v : \text{Type} \times \text{Type} \to \text{Type}$)
$w : S(\int)$
$w' : S(w)$

4.1 Grammar and Judgements (Attempt 1)

Grammar:

$k ::= \text{Type} \mid k \to k \mid k \times k \mid S(c)$
$c ::= \ldots$

Judgements:

$$\frac{}{\Gamma \vdash c : S(c)} \qquad \frac{\Gamma \vdash c : S(c)}{\Gamma \vdash c \equiv c' : \text{Type}} \qquad \frac{\Gamma \vdash c : \text{Type}}{\Gamma \vdash S(c) : \text{kind}}$$

Signature for `list`.

```
sig
  .
  .
  .
  type 'a s = 'a list
  type 'a t
end
```

So we have $t : \text{Type} \to \text{Type}$.
How do we represent 'a s? Is $s : \text{Type} \to S(\alpha)$? But then what's α.

4.2 Dependent Kinds (Grammar)

$k ::= \text{Type} \mid \Pi\alpha : k \,.\, k \mid \Sigma\alpha : k \,.\, k \mid S(c)$
$c ::= \ldots$

Note: Π is also known as "dependent product"

Σ is also known as "dependent sum" (but also sometimes as "dependent product").

To avoid confusion, we name Π "dependent function (spaces)".

Now, we have $s : \Pi\alpha : \text{Type} \,.\, S(list\,\alpha)$.

New judgements we need to be able to make:
$\Gamma \vdash k : \text{kind}$
$\Gamma \vdash k \equiv k' : kind$
$\Gamma \vdash k \leq k'$
$\Gamma \vdash c : k$
$\Gamma \vdash c \equiv c' : k$
$\Gamma \vdash e : \tau$

Note: $S(f) \leq \text{Type}$

4.3 $\Gamma \vdash k : \text{kind}$

$$\frac{}{\Gamma \vdash \text{Type} : \text{kind}} \qquad \frac{\Gamma \vdash c : \text{Type}}{\Gamma \vdash S(c) : \text{kind}} \qquad \frac{\Gamma \vdash k_1 : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 : \text{kind}}$$

$$\frac{\Gamma \vdash k_1 : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash \Sigma\alpha : k_1 \,.\, k_2 : \text{kind}}$$

4.4 $\Gamma \vdash k \equiv k' : \text{kind}$

$$\frac{\Gamma \vdash k : \text{kind}}{\Gamma \vdash k \equiv k : \text{kind}} \qquad \frac{\Gamma \vdash k_1 \equiv k_2 : \text{kind}}{\Gamma \vdash k_2 \equiv k_1 : \text{kind}} \qquad \frac{\Gamma \vdash k_1 \equiv k_2 : \text{kind} \qquad \Gamma \vdash k_2 \equiv k_2 : \text{kind}}{\Gamma \vdash k_1 \equiv k_2 : \text{kind}} \qquad \frac{\Gamma \vdash c \equiv c' : \text{Type}}{\Gamma \vdash S(c) \equiv S(c') : \text{kind}}$$

$$\frac{\Gamma \vdash k_1 \equiv k_1' : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 \equiv k_2' : \text{kind}}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 \equiv \Pi\alpha : k_1' \,.\, k_2' : \text{kind}} \qquad \frac{\Gamma \vdash k_1 \equiv k_1' : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 \equiv k_2' : \text{kind}}{\Gamma \vdash \Sigma\alpha : k_1 \,.\, k_2 \equiv \Sigma\alpha : k_1' \,.\, k_2' : \text{kind}}$$

Note: for the latter two, keep $\Pi\alpha : k_1 \,.\, k_2 \overset{?}{\equiv} \Pi\alpha' : k_1' \,.\, k_2'$ in mind

4.5 $\Gamma \vdash \alpha : k$

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha : k} \qquad \frac{\Gamma \vdash c_1 : \text{Type} \qquad \Gamma \vdash c_2 : \text{Type}}{\Gamma \vdash c_1 \to c_2 : \text{Type}} \qquad \frac{\Gamma \vdash k : kid \qquad \Gamma, \alpha : k \vdash c : \text{Type}}{\Gamma \vdash \forall\alpha : k \,.\, c : \text{Type}}$$

$$\frac{\Gamma \vdash k_1 : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash c : k_2}{\Gamma \vdash \lambda\alpha : k_1 \,.\, c : \Pi\alpha : k_1 \,.\, k_2} \qquad \frac{\Gamma \vdash c_1 : \Pi\alpha : k \,.\, k' \qquad \Gamma \vdash c_2 : k}{\Gamma \vdash c_1\ c_2 : [c_1 \,/\, \alpha]k'}$$

$$\frac{\Gamma \vdash c_1 : k_2 \qquad \Gamma \vdash c_2 : [c_1 \,/\, \alpha]k_2 \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash \langle c_1, c_2 \rangle : \Sigma\alpha : k_2 \,.\, k_2} \qquad \frac{\Gamma \vdash c : \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_1 c : k_1} \qquad \frac{\Gamma \vdash c : \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_2 c : [\pi_1 c \,/\, \alpha]k_2}$$

$$\frac{\Gamma \vdash c : k \qquad \Gamma \vdash k \leq k'}{\Gamma \vdash c : k'}$$

Additional Judgements If $\vdash \Gamma\,\mathrm{ok}$ and $\Gamma \vdash c : k$, then $\Gamma \vdash k : \mathrm{kind}$.
If $\vdash \Gamma\,\mathrm{ok}$ and $\Gamma \vdash k_1 \equiv k_2$, then $\Gamma \vdash k_1, k_2\,\mathrm{kind}$.
If $\vdash \Gamma\,\mathrm{ok}$ and $\Gamma \vdash k_1 \leq k_2$, then $\Gamma \vdash k_1, k_2\,\mathrm{kind}$.

$$\frac{\Gamma \vdash c : \mathrm{Type}}{\Gamma \vdash c : S(c)}$$

Sub-kinding:

$$\frac{\Gamma \vdash k \equiv k' : \mathrm{kind}}{\Gamma \vdash k \leq k'} \qquad \frac{\Gamma \vdash k_1 \leq k_2 \qquad \Gamma \vdash k_2 \leq k_3}{\Gamma \vdash k_1 \leq k_3} \qquad \frac{\Gamma \vdash c : \mathrm{Type}}{\Gamma \vdash S(c) \leq \mathrm{Type}} \qquad \frac{\Gamma \vdash c \equiv c' : \mathrm{Type}}{\Gamma \vdash S(c) \leq S(c')}$$

$$\frac{\Gamma \vdash k_1' \leq k_1 \qquad \Gamma, \alpha : k_1' \vdash k_2 \leq k_2' \qquad \Gamma, \alpha : k_1 \vdash k_2 : \mathrm{kind}}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 \leq \Pi\alpha : k_1' \,.\, k_2'}$$

$$\frac{\Gamma \vdash k_1 \leq k_1' \qquad \Gamma, \alpha : k_1 \vdash k_2 \leq k_2' \qquad \Gamma, \alpha : k_1' \vdash k_2' : \mathrm{kind}}{\Gamma \vdash \Sigma\alpha : k_1 \,.\, k_2 \leq \Sigma\alpha : k_1' \,.\, k_2'}$$

Note: Something about contravariance for 1st condition. Π contravariant the same way arrow is contravariant. Covariance for 2nd condition. This is for Π.

4.6 $\Gamma \vdash c \equiv c : k$

$$\frac{\Gamma \vdash c : k}{\Gamma \vdash c \equiv c : k} \qquad \frac{\Gamma \vdash c_1 \equiv c_2 : k}{\Gamma \vdash c_2 \equiv c_1 : k} \qquad \frac{\Gamma \vdash c_1 \equiv c_2 : k \qquad \Gamma \vdash c_2 \equiv c_3 : k}{\Gamma \vdash c_1 \equiv c_3 : k}$$

$$\frac{\Gamma \vdash c_2 : k \qquad \Gamma, \alpha : k \vdash c_1 : k'}{\Gamma \vdash (\lambda\alpha : k \,.\, c_1)\ c_2 \equiv [c_2 \,/\, \alpha]c_1 : [c_2 \,/\, \alpha]k'} \qquad \frac{\Gamma \vdash c_1 : k_1 \qquad \Gamma \vdash c_2 : k_2}{\Gamma \vdash \pi_i\langle c_1, c_2\rangle \equiv c_i : k_i} \qquad \frac{\Gamma \vdash c : S(c')}{\Gamma \vdash c \equiv c' : S(c')}$$

$$\frac{\Gamma \vdash c_1 \equiv c_2 : k \qquad \Gamma \vdash k \leq k'}{\Gamma \vdash c_1 \equiv c_2 : k'}\,\star \qquad \frac{\Gamma \vdash c \equiv c' : \mathrm{Type}}{\Gamma \vdash c \equiv c' : S(c)}\,\star \qquad \frac{\Gamma \vdash k_1 \equiv k_1' : \mathrm{kind} \qquad \Gamma, \alpha : k_1 \vdash c \equiv c' : k_2}{\Gamma \vdash \lambda\alpha : k_1 \,.\, c \equiv \lambda\alpha : k_1' \,.\, c' : \Pi\alpha : k_1 \,.\, k_2}$$

$$\frac{\Gamma \vdash c_1 \equiv c_1' : \Pi\alpha : k \,.\, k' \qquad \Gamma \vdash c_2 \equiv c_2' : k}{\Gamma \vdash c_1\ c_2 \equiv c_1'\ c_2' : [c_2 \,/\, \alpha]k'}$$

5 Sub-Typing

$\tau \leq \tau'$ means you can use a τ whrever a τ' is expected.

$$\frac{\Gamma \vdash e : \tau \qquad \tau \leq \tau'}{\Gamma \vdash e : \tau'} \qquad \frac{\tau_1 \leq \tau_1' \qquad \tau_2 \leq \tau_2'}{\Gamma \vdash \tau \times \tau_2 \leq \tau_1' \times \tau_2'} \qquad \frac{\tau_1' \leq \tau_1 \qquad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'}$$

Note 1: This is a case of covariance on both sides.
Note 2: This is contravariant on the left and covariant on the right.

$\mathcal{N} \leq \mathcal{R}$.
Assume we have $f : \mathcal{N} \to \mathcal{N}$.
$f \not: \mathcal{R} \to \mathcal{R}$.

Assume we have $f : \mathcal{R} \to \mathcal{R}$.
Contravariance: $f : \mathcal{N} \to \mathcal{R}$.

$$\frac{\tau \equiv \tau' : \text{Type}}{\text{ref}(\tau) \leq \text{ref}(\tau')}$$

ref(τ) is neither covariant nor contravariant. Called "invariant". (Poorly named, but it's what's used in literature.)

$$\frac{\Gamma \vdash k_1' \leq k_1 \qquad \Gamma, \alpha : k_1' \vdash k_2 \leq k_2' \qquad \Gamma, \alpha : k_1 \vdash k_2 : \mathit{kind}}{\Gamma \vdash \Pi\alpha : k_1 \;.\; k_2 \leq \Pi\alpha : k_1' \;.\; k_2'}$$

$$\frac{\Gamma \vdash k_1' \leq k_1 \qquad \Gamma, \alpha : k_1' \vdash k_2 \leq k_2' \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash \Sigma\alpha : k_1 \;.\; k_2 \leq \Sigma\alpha : k_1' \;.\; k_2'} \qquad \frac{\Gamma \vdash c : S(c')}{\Gamma \vdash c \equiv c' : S(c')} \qquad \frac{\Gamma \vdash c : S(c')}{\Gamma \vdash c \equiv c' : \text{Type}}$$

$$\frac{\Gamma \vdash c \equiv c' : \text{Type}}{\Gamma \vdash c : c' : S(c)}$$

Note 1: Sound, but not what we want.

More compatiblity rules.

$$\frac{\Gamma \vdash c_1 \equiv c_1' : k_1 \qquad \Gamma \vdash c_2 \equiv c_2' : [c_1 \;/\; \alpha]k_2 \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash \langle c_1, c_2 \rangle \equiv \langle c_1', c_2' \rangle : \Sigma\alpha : k_1 \;.\; k_2} \qquad \frac{\Gamma \vdash c \equiv c' : \Sigma\alpha : k_1 \;.\; k_2}{\Gamma \vdash \pi_1 c \equiv \pi_1 c' : k_1}$$

$$\frac{\Gamma \vdash c \equiv c' : \Sigma\alpha : k_1 \;.\; k_2}{\Gamma \vdash \pi_2 c \equiv \pi_2 c' : [\pi_1 c \;/\; \alpha]k_2} \qquad \frac{\Gamma \vdash c_1 \equiv c_1' : \text{Type} \qquad \Gamma \vdash c_2 \equiv c_2' : \text{Type}}{\Gamma \vdash c_1 \to c_2 \equiv c_1' \to c_2' : \text{Type}}$$

$$\frac{\Gamma \vdash k \equiv k' : \text{kind} \qquad \Gamma, \alpha : k \vdash c \equiv c' : \text{Type}}{\Gamma \vdash \forall\alpha : k \;.\; c \equiv \forall\alpha : k' \;.\; c' : \text{Type}}$$

Rules for extentionality.

$$\frac{\Gamma, \alpha : k_1 \vdash c\ \alpha \equiv c'\ \alpha : k_2 \qquad \Gamma \vdash c : \Pi alpha : k_1\ .\ k_2' \qquad \Gamma \vdash c' : \Pi alpha : k_1\ .\ k_2''}{\Gamma \vdash c \equiv c' : \Pi alpha : k_1\ .\ k_2}$$

$$\frac{\Gamma, \alpha : k_1 \vdash c\ \alpha \equiv c'\ \alpha : k_2 \qquad \Gamma \vdash c \equiv c' : \Pi alpha : k_1\ .\ k_2'}{\Gamma \vdash c \equiv c' : \Pi\alpha : k_1\ .\ k_2}$$

$$\frac{\Gamma \vdash \pi_1 c \equiv \pi_1 c' : k_1 \qquad \Gamma \vdash \pi_2 c \equiv \pi_2 c' : [\pi_1 c\ /\ \alpha]k_2 \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash c \equiv c' : \Sigma alpha : k_1\ .\ k_2}$$

Note 1: We only need this for proofs (regularity). We can safely ignore this.

We have no way of dealing with $\mathcal{S}(c : k)$. So instead of redefining everything, treat it as a macro following the following rules:
$\mathcal{S}(c : \text{Type}) = \mathcal{S}(c)$
$\mathcal{S}(c : \Pi\alpha : k_1\ .\ k_2) = \Pi\alpha : k_1\ .\ \mathcal{S}(c\ \alpha : k_2)$
$\mathcal{S}(c : \mathcal{S}(c')) = \mathcal{S}(c)$ (note here, $c \equiv c'$, so we can use either, but it's easier for us to use c)
$\mathcal{S}(c : \Pi\alpha : k_1\ .\ k_2) = \Sigma\alpha : \mathcal{S}(\pi_1 c : k_1)\ .\ \mathcal{S}(\pi_2 c : k_2)$
OR $\mathcal{S}(\pi_1 c : k_1) \times \mathcal{S}(\pi_2 c : [\pi_1 c\ /\ \alpha]k_2)$
We use the 2nd because it's nicer when not working without theory. The first is more theoretic, the second is syntactic.

1. If $\Gamma \vdash c : k$, then $\Gamma \vdash c : \mathcal{S}(c : k)$
2. If $\Gamma \vdash c : \mathcal{S}(c' : k)$, then $\Gamma \vdash c \equiv c' : k$

But the first doesn't hold. So let's make it hold. Add "declarative" rules:

$$\frac{\Gamma \vdash k_1 : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash c\ \alpha : k_2}{\Gamma \vdash c : \Pi\alpha : k_1\ .\ k_2} \qquad \frac{\Gamma \vdash \pi_1 c : k_2 \qquad \Gamma \vdash \pi_1 c : [\pi_1 c\ /\ \alpha]k_2 \qquad \Gamma, \alpha : k_1 \vdash k_2 : \text{kind}}{\Gamma \vdash c : \Sigma\alpha : k_1\ .\ k_2}$$

Notes on definitional equivalence:
$\alpha : \text{Type} \vdash \alpha \not\equiv \text{int} : \text{Type}$
$\alpha : \mathcal{S}(\text{int}) \vdash \alpha \equiv \text{int} : \text{Type}$
$\vdash \lambda\alpha : \text{Type}\ .\ \alpha \not\equiv \lambda\alpha : \text{Type}\ .\ \text{int} : \text{Type} \to \text{Type}$
$\vdash \lambda\alpha : \text{Type}\ .\ \alpha \not\equiv \lambda\alpha : \text{Type}\ .\ \text{int} : \mathcal{S}(\text{int}) \to \text{Type}$
$\beta : (\text{Type} \to \text{Type}) \to \text{Type} \vdash \beta(\lambda\alpha : \text{Type}\ .\ \alpha \not\equiv \beta(\lambda\alpha : \text{Type}\ .\ \text{int} : \text{Type}$
$\beta : (\mathcal{S}(\text{int}) \to \text{Type}) \to \text{Type} \vdash \beta(\lambda\alpha : \text{Type}\ .\ \alpha \equiv \beta(\lambda\alpha : \text{Type}\ .\ \text{int} : \text{Type}$
$\text{Type} \to \text{Type} \leq \mathcal{S}(\text{int}) \to \text{Type}$

5.1 Algorithm for Equivalence Checking

$$\frac{\Gamma, \alpha : k_1 \vdash c\ \alpha \Leftrightarrow c'\ \alpha : k_2}{\Gamma \vdash c \Leftrightarrow c' : \Pi\alpha : k_1\ .\ k_2} \qquad \frac{\Gamma \vdash \pi_1 c \Leftrightarrow \pi_2 c' : k_1 \Gamma \vdash \pi_2 c \Leftrightarrow \pi_2 c' : [\pi_1 c\ /\ \alpha]k_2}{\Gamma \vdash c \Leftrightarrow c' : \Sigma\alpha : k_1\ .\ k_2}$$

$$\frac{\Gamma \vdash c_1 \Downarrow c_1' \qquad \Gamma \vdash c_2 \Downarrow c_2' \qquad \Gamma \vdash c_1' \leftrightarrow c_2' : \text{Type}}{\Gamma \vdash c_1 \Leftrightarrow c_2 : \text{Type}} \qquad \frac{\Gamma \vdash c \rightsquigarrow c' \qquad \Gamma \vdash c' \Downarrow c''}{\Gamma \vdash c \Downarrow c''} \qquad \frac{\Gamma \vdash c \not\rightsquigarrow}{\Gamma \vdash c \Downarrow c}$$

$$\frac{}{\Gamma \vdash (\lambda\alpha : k\ .\ c_1)\ c_2 \rightsquigarrow [c_2\ /\ \alpha]c_1} \qquad \frac{\Gamma \vdash c_1 \rightsquigarrow c_1'}{\Gamma \vdash c_1\ c_2 \rightsquigarrow c_1'\ c_2} \qquad \frac{}{\Gamma \vdash \pi_i\langle c_1, c_2\rangle \rightsquigarrow c_i} \qquad \frac{\Gamma \vdash c \rightsquigarrow c'}{\Gamma \vdash pi_i c \rightsquigarrow pi_i c'}$$

$$\frac{\Gamma \vdash p \uparrow \mathcal{S}(c)}{\Gamma \vdash p} \qquad \frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha \uparrow k} \qquad \frac{\Gamma \vdash p \uparrow \Pi\alpha : k_1\ .\ k_2}{\Gamma \vdash p\ c \uparrow [c\ /\ \alpha]k_2} \qquad \frac{\Gamma \vdash p \uparrow \Sigma\alpha : k_1\ .\ k_2}{\Gamma \vdash \pi_1 p \uparrow k_1} \qquad \frac{\Gamma \vdash p \uparrow \Sigma\alpha : k_1\ .\ k_2}{\Gamma \vdash \pi_2 p \uparrow [\pi_1 p\ /\ \alpha]k_2}$$

$$\frac{\Gamma \vdash p \uparrow S(c)}{\Gamma \vdash p \rightsquigarrow c}$$

Example:

$$\dfrac{\dfrac{\dots \vdash (\lambda\alpha : \text{Type}\alpha\alpha \rightsquigarrow \alpha \qquad \dfrac{\dfrac{\alpha : S(\text{int}) \vdash \alpha \uparrow S(\text{int})}{\alpha : S(\text{int}) \vdash \alpha \rightsquigarrow \text{int}} \qquad \dfrac{\dots \vdash \text{int} \not\rightsquigarrow}{\dots \vdash \text{int} \Downarrow \text{int}}}{\dots \vdash \alpha \Downarrow \text{int}}}{\alpha : S(\text{int}) \vdash (\lambda\alpha : \text{Type} \,.\, \alpha)\alpha \Downarrow} \qquad \dfrac{}{\alpha : S(\text{int}) \vdash (\lambda\alpha : \text{Typeint})\alpha \Downarrow \text{int}} \qquad \dfrac{}{\alpha : S(\text{int}) \vdash \text{int} \leftrightarrow \text{int} : \text{Type}}}{\dfrac{\alpha : S(\text{int}) \vdash (\lambda\alpha : \text{Type} \,.\, \alpha)\alpha \Leftrightarrow (\lambda\alpha : \text{Type} \,.\, \text{int})\alpha \Leftrightarrow}{\vdash \lambda\alpha : \text{Type} \,.\, \alpha \Leftrightarrow \lambda\alpha : \text{Type} \,.\, \text{int} : S(\text{int}) \to \text{Type}}}$$

One final rule:

$$\frac{}{\Gamma \vdash c_1 \Leftrightarrow c_2 : S(c)}$$

The precondition is that both c_1 and c_2 belong to $S(c)$, meaning they are equivalent to c and by transitivity, equivalent to each other.

Some rules that we will never use:

$$\frac{}{\Gamma \vdash c_1 \to c_2 \uparrow \text{Type}} \qquad \frac{}{\Gamma \vdash \forall\alpha : k \,.\, c \uparrow \text{Type}}$$

Structural rules:

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha \leftrightarrow \alpha : k} \qquad \frac{\Gamma \vdash p \leftrightarrow p' : \Pi\alpha : k_1 \,.\, k_2 \qquad \Gamma \vdash c \Leftrightarrow c' : k_1}{\Gamma \vdash p\ c \leftrightarrow p'\ c' : [c \,/\, \alpha]k_2} \qquad \frac{\Gamma \vdash p \leftrightarrow p' : \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_1 p \leftrightarrow \pi_1 p' : k_1}$$

$$\frac{\Gamma \vdash p \leftrightarrow p' : \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_1 p \leftrightarrow \pi_1 p' : [\pi_1 p \,/\, \alpha]k_2} \qquad \frac{\Gamma \vdash c_1 \Leftrightarrow c_1' : \text{Type} \qquad \Gamma \vdash c_2 \Leftrightarrow c_2' : \text{Type}}{\Gamma \vdash c_1 \to c_2 \leftrightarrow c_1' \to c_2' : \text{Type}}$$

$$\frac{\Gamma \vdash k \Leftrightarrow k' : \text{kind} \qquad \Gamma, \alpha : k \vdash c \Leftrightarrow c' : \text{Type}}{\Gamma \vdash \forall\alpha : k \,.\, c \leftrightarrow \forall\alpha : k' \,.\, c' : \text{Type}}$$

If $\Gamma \vdash c \leftrightarrow c' : k$ then $\Gamma \vdash c \uparrow k$ also $\exists k' \,.\, \Gamma \vdash c' \uparrow k'$ and $\Gamma \vdash k \equiv k' : \text{kind}$

Structural comparison:

$$\frac{}{\Gamma \vdash \text{Type} \Leftrightarrow \text{Type} : \text{kind}} \quad \frac{\Gamma \vdash c \Leftrightarrow c' : \text{Type}}{\Gamma \vdash S(c) \Leftrightarrow S(c') : \text{kind}} \qquad \frac{\Gamma \vdash k_1 \Leftrightarrow k_1' : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 \Leftrightarrow k_2' : \text{kind}}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 \Leftrightarrow \Pi\alpha : k_1' \,.\, k_2'}$$

$$\frac{\Gamma \vdash k_1 \Leftrightarrow k_1' : \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 \Leftrightarrow k_2' : \text{kind}}{\Gamma \vdash \Sigma\alpha : k_1 \,.\, k_2 \Leftrightarrow \Sigma\alpha : k_1' \,.\, k_2'}$$

$\Gamma \vdash k \trianglelefteq k'$

$$\frac{}{\Gamma \vdash \text{Type} \trianglelefteq \text{Type}} \qquad \frac{}{\Gamma \vdash S(c) \trianglelefteq \text{Type}} \qquad \frac{\Gamma \vdash c \Leftrightarrow c' : \text{Type}}{\Gamma \vdash S(c) \trianglelefteq S(c')} \qquad \frac{\Gamma \vdash k_1' \trianglelefteq k_1 \qquad \Gamma, \alpha : k_1' \vdash k_2 \trianglelefteq k_2'}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 \trianglelefteq \Pi\alpha : k_1' \,.\, k_2'}$$

$$\frac{\Gamma \vdash k_1 \trianglelefteq k_1' \qquad \Gamma, \alpha : k_1 \vdash k_2 \trianglelefteq k_2'}{\Gamma \vdash \Sigma\alpha : k_1 \,.\, k_2 \trianglelefteq \Sigma\alpha : k_1' \,.\, k_2'}$$

$\Gamma \vdash k \Leftarrow \text{kind}$

$$\frac{}{\Gamma \vdash \text{Type} \Leftarrow \text{kind}} \qquad \frac{\Gamma \vdash c \Leftarrow \text{Type}}{\Gamma \vdash S(c) \Leftarrow \text{kind}} \qquad \frac{\Gamma \vdash k_1 \Leftarrow \text{kind} \qquad \Gamma, \alpha : k_1 \vdash k_2 \Leftarrow \text{kind}}{\Gamma \vdash \Pi\alpha : k_1 \,.\, k_2 \Leftarrow \text{kind}}$$

Suppose $\vdash \Gamma\,\text{ok}$. Then:

<u>S</u>oundness

- If $\Gamma \vdash c_1, c_2 : k$ and $\Gamma \vdash c_1 \Leftrightarrow c_2 : k$ then $\Gamma \vdash c_1 \equiv c_2 : k$
- If $\Gamma \vdash k_1, k_2 : \text{kind}$ and $\Gamma \vdash k_1 \Leftrightarrow k_2 : \text{kind}$ then $\Gamma \vdash k_1 \equiv k_2 : \text{kind}$
- If $\Gamma \vdash k_1, k_2 : \text{kind}$ and $\Gamma \vdash k_1 \trianglelefteq k_2$ then $\Gamma \vdash k_1 \leq k_2$
- If $\Gamma \vdash k \Leftarrow \text{kind}$ then $\Gamma \vdash k : \text{kind}$
- If $\Gamma \vdash c \Rightarrow k$ then $\Gamma \vdash c : k$

<u>C</u>ompleteness

- If $\Gamma \vdash c_1 \equiv c_2 : k$ then $\Gamma \vdash c_1 \Leftrightarrow c_2 : k$
- If $\Gamma \vdash k_1 \equiv k_2 : \text{kind}$ then $\Gamma \vdash k_1 \Leftrightarrow k_2 : \text{kind}$
- If $\Gamma \vdash k_1 \leq k_2$ then $\Gamma \vdash k_1 \trianglelefteq k_2$
- If $\Gamma \vdash k : \text{kind}$ then $\Gamma \vdash k \Leftarrow \text{kind}$
- If $\Gamma \vdash c : k$ then $\Gamma \vdash c \Rightarrow k'$ and $\Gamma \vdash k' \leq S(c : k)$

TODO: principle type
TODO: principle kind is a subkind of every other kind

Checking principle...
$\Gamma \vdash c \Rightarrow k$

$$\frac{\Gamma \vdash c \Rightarrow k' \qquad \Gamma \vdash k' \trianglelefteq k}{\Gamma \vdash c \Leftarrow k}$$

$$\frac{\Gamma(\alpha) = k}{\Gamma \vdash \alpha \Rightarrow S(\alpha : k)} \qquad \frac{\Gamma \vdash k \Leftarrow \text{kind} \qquad \Gamma, \alpha : k \vdash c \Rightarrow k'}{\Gamma \vdash \lambda\alpha : k \,.\, c \Rightarrow \Pi\alpha : k \,.\, k'} \qquad \frac{\Gamma \vdash c_1 \Rightarrow \Pi\alpha : k \,.\, k' \qquad \Gamma \vdash c_2 \Leftarrow k}{\Gamma \vdash c_1\ c_2 \Rightarrow [c_2 \,/\, \alpha]k'}$$

$$\frac{\Gamma \vdash c_1 \Rightarrow k_1 \qquad \Gamma \vdash c_2 \Rightarrow k_2}{\Gamma \vdash \langle c_1, c_2 \rangle \Rightarrow k_1 \times k_2} \qquad \frac{\Gamma \vdash c \Rightarrow \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_1 c \Rightarrow k_1} \qquad \frac{\Gamma \vdash c \Rightarrow \Sigma\alpha : k_1 \,.\, k_2}{\Gamma \vdash \pi_2 c \Rightarrow [\pi_1 c \,/\, \alpha]k_2}$$

$$\frac{\Gamma \vdash c_1 \Leftarrow \text{Type} \qquad \Gamma \vdash c_2 \Leftarrow \text{Type}}{\Gamma \vdash c_1 \to c_2 \Rightarrow S(c_1 \to c_2)} \qquad \frac{\Gamma \vdash k \Leftarrow \text{kind} \qquad \Gamma, \alpha : k \vdash c \Leftarrow \text{Type}}{\Gamma \vdash \forall\alpha : k \,.\, c \Rightarrow S(\forall\alpha : k \,.\, c)}$$

6 Checking Expressions

$\Gamma \vdash e \Rightarrow \tau$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x \Rightarrow \tau} \qquad \frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \quad \Gamma \vdash \tau_1 \Downarrow \tau \to \tau' \quad \Gamma \vdash e_2 \Leftarrow \tau}{\Gamma \vdash e_1\ e_2 \Rightarrow \tau'}$$

7 Type-Directed Translation / Syntax-Directed Translation

A more accurate name: "Typing-derivation-directed translation". We proceed by the analysis of the typing derivation of the rules.
Let's represent the source and target languages in different colors, to indicate that they are different.

Property:
$\Gamma \vdash e : \tau$ if and only if $\exists e\ .\ \Gamma \vdash e : \tau \rightsquigarrow e$.

We also want:
If $\Gamma \vdash e : \tau \rightsquigarrow e$, something like $\Gamma \vdash e : \tau$.
But we have no concept of Γ or τ or its derivations.
Instead:
Property:
If $\Gamma \vdash e : \tau \rightsquigarrow e$ and $\tau \rightsquigarrow \tau$ and $\Gamma \rightsquigarrow \Gamma$, then $\Gamma \vdash e : \tau$.
Why not $\Gamma \vdash \tau : \text{Type} \rightsquigarrow \tau$.

Simply, we'll use " If $\Gamma \vdash e : \tau \rightsquigarrow e$ then $\Gamma \vdash e : \tau$

7.1 Coherence

For Terms:
Suppose $\Gamma \vdash e : \tau \rightsquigarrow e$ and $\Gamma \vdash e : \tau \rightsquigarrow e'$.
$\Gamma \vdash e \cong e' : \tau$.
This is too hard to even define, this is left to graduate courses. We aspire to it but it's too much of a pain to actually do.

For Types:
Suppose $\Gamma \vdash c : k \rightsquigarrow c$ and $\Gamma \vdash c : k \rightsquigarrow c'$.
Then,
$\Gamma \vdash c \equiv c' : k$.
This is not an aspiration, we cannot live without this.
The 2nd property above can't even be made without this, but it doesn't have to be kind directed. And instead, we'll just make it syntax directed, which will trivially prove that the two are equivalent.

7.2 Definition of e

$$\begin{aligned}
\alpha &= \alpha \\
\tau_1 \times \tau_2 &= \tau_1 \times \tau_2 \\
\tau_1 \to \tau_2 &= \text{unit} \to \tau_1 \to \tau_2 \\
&\cdots \\
\epsilon &= \epsilon \\
\Gamma, x : \tau &= \Gamma, x : \tau \\
\Gamma, \alpha : k &= \Gamma, \alpha : k
\end{aligned}$$

Convoluted example:

$$\frac{\Gamma \vdash \tau : \text{Type} \qquad \Gamma, x : \tau \vdash e : \tau \rightsquigarrow e}{\Gamma \vdash \lambda x : \tau \,.\, e : \tau \to \tau' \rightsquigarrow \lambda z : \text{unit} \,.\, \lambda x : \tau \,.\, e}$$

NOTE: the right part (after $\rightsquigarrow$) indicates the need to shift in terms of debruijn indices.

More:

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \rightsquigarrow {e_1}^{:\tau_1 \to \tau_2 = \text{unit} \to \tau \to \tau'} \qquad \Gamma \vdash e_2 : \tau \rightsquigarrow {e_2}^{:\tau}}{\Gamma \vdash e_1\ e_2 : \tau' \rightsquigarrow e_1 <> e_2}$$

More (only well typed things translate):

$$\frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \rightsquigarrow e_1 \qquad \Gamma \vdash e_1 \Downarrow \tau \to \tau' \qquad \Gamma \vdash e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \qquad \Gamma \vdash \tau_2 \Leftrightarrow \tau : \text{Type}}{\Gamma \vdash e_1\ e_2 \Rightarrow \tau' \rightsquigarrow e_1 <> e_2}$$

7.3 $\Gamma \vdash e : \tau \rightsquigarrow e$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \rightsquigarrow x} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow e_1 \qquad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2 \rightsquigarrow \langle e_1, e_2 \rangle} \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow e}{\Gamma \vdash \pi_1 e : \tau_1 \rightsquigarrow \pi_1 e}$$

8 More Things

$$\frac{\Gamma \vdash c : 1 \qquad \Gamma \vdash c' : 1}{\Gamma \vdash c \equiv c' : 1} \qquad\qquad \frac{}{\Gamma \vdash * : 1}$$

$$\frac{\Gamma \vdash c : k \qquad \Gamma \vdash e : [c \,/\, \alpha]\tau \qquad \Gamma, \alpha : k \vdash \tau : \text{Type}}{\Gamma \vdash (\texttt{pack}[c, e]\, as\, \exists \alpha : k \,.\, \tau) : \exists \alpha : k \,.\, \tau}$$

$$\frac{\Gamma \vdash e_1 : \exists \alpha : k \,.\, \tau \qquad \Gamma, \alpha : k, x : \tau \vdash e_2 : \tau' \qquad \Gamma \vdash \tau' : \text{Type}}{\Gamma \vdash \texttt{unpack}[\alpha, x] = e_1 ine_2 : \tau'}$$

$$\frac{\Gamma \vdash \tau : \text{Type}}{\Gamma \vdash \texttt{newtag}[\tau] : \texttt{tag}t} \qquad\qquad \frac{\Gamma \vdash e_1 : \texttt{tag} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \texttt{tag}(e_1, e_2) : \texttt{exn}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{tag} \qquad \Gamma \vdash e_2 : \texttt{exn} \qquad \Gamma, x : e \vdash e_3 : \tau' \qquad \Gamma \vdash e_4 : \tau'}{\Gamma \vdash \texttt{iftag}(e_1, e_2, x \,.\, e_3, e_4) : \tau'}$$

$$\frac{\Gamma \vdash e_1 : \exists \alpha : k \,.\, \tau \qquad \Gamma, \alpha : k, x : \tau \vdash e_2 : \tau' \qquad \Gamma \vdash \tau : \text{Type}}{\Gamma \vdash \texttt{unpack}[\alpha, x] = e_1 ine_2 : \tau'}$$

$$\frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \qquad \Gamma \vdash c : k \qquad \Gamma \vdash e_1 \Downarrow \exists \alpha : k \,.\, \tau \qquad \Gamma, \alpha : k, x : \tau \vdash e_2 \Rightarrow \tau' \qquad \Gamma, \alpha : k \vdash \tau \Leftrightarrow [c \,/\, \alpha]\tau' : \text{Type}}{\Gamma \vdash \texttt{unpack}[\alpha, x] = e_1 ine_2 \Rightarrow [c \,/\, \alpha]\tau'}$$

9 Continuation-Passing Style (CPS)

- control-flow is explicit
- name all intermediate results
- reify control-flow (continuations) as data

The first two are often called “monadic form” or in literature, “A-normal form” (or by Harper, 2/3 CPS).

9.1 Target Language

Still have k, c, and now we have expressions e (which do not return) and values v.

$k ::= \dots$
$c ::= \cdots \mid \cancel{\tau \to \tau} \mid \cancel{\forall \alpha : k \,.\, \tau} \mid \neg\tau$
$e ::= v\ v \mid \texttt{unpack}[\alpha, x] = vine \mid \texttt{let}\ x = \pi_i v\ \texttt{in}\ e\ \texttt{end} \mid \texttt{let}\ x = v\ \texttt{in}\ e\ \texttt{end} \mid \texttt{halt} \mid \dots v \quad ::= x \mid \lambda x : \tau \,.\, e \mid \texttt{pack}[c, v] as \exists\alpha$

Judgements:
$\Gamma \vdash v : \tau$
$\Gamma \vdash e : 0$
(e does not return, so we use ‘0’ for ‘OK’)

$$\frac{\Gamma \vdash \tau : \text{Type} \qquad \Gamma, x : \tau \vdash e : 0}{\Gamma \vdash \lambda x : \tau \,.\, e : \neg\tau} \qquad \frac{\Gamma \vdash v_1 : \neg\tau \qquad \Gamma \vdash v_2 : \tau}{\Gamma \vdash v_1\ v_2 : 0}$$

$$\frac{\Gamma \vdash c : k \qquad \Gamma \vdash v : [c \,/\, \alpha]\tau \qquad \Gamma, \alpha : k \vdash \tau : \text{Type}}{\Gamma \vdash \texttt{pack}[c, v] as \exists\alpha : k \,.\, \tau : \exists\alpha : k \,.\, \tau} \qquad \frac{\Gamma \vdash v : \exists\alpha : k \,.\, \tau \Gamma, \alpha : k, x : \tau \vdash e : 0}{\Gamma \vdash \texttt{unpack}[\alpha, x] = v \in e : 0}$$

$$\frac{\Gamma \vdash v_i : \tau_i \qquad (\forall i \in [n])}{\Gamma \vdash \langle v_1, \dots v_2 \rangle : x[\tau, \dots \tau_n]} \qquad \frac{\Gamma \vdash v : x[\tau_0, \dots \tau_{n-1} \qquad \Gamma, x : \tau_i \vdash e : 0}{\Gamma v \texttt{let}\ x = \pi_i v\ \texttt{in}\ e\ \texttt{end} : 0} \qquad \frac{\Gamma \vdash v : \tau \qquad \Gamma, x : \tau \vdash e : 0}{\Gamma \vdash \texttt{let}\ x = v\ \texttt{in}\ e\ \texttt{end} : 0}$$

$$\frac{}{\Gamma \vdash \texttt{halt} : 0}$$

9.2 Translation

(NOTE: this is syntax-directed)

$$
\begin{aligned}
T &= T \\
\Pi\alpha : k_1 \,.\, k_2 &= \Pi\alpha : k_1 \,.\, k_2 \\
S(c) &= S(c) \\
1 &= 1 \\
\alpha &= \alpha \\
\lambda\alpha : k \,.\, c &= \lambda\alpha : k \,.\, c \\
c_1\ c_2 &= c_1\ c_2 \\
\langle c_1, c_2 \rangle &= \langle c_1, c_2 \rangle \\
\pi_1 c &= \pi_1 c \\
\pi_2 c &= \pi_2 c
\end{aligned}
$$

$$
\begin{aligned}
\tau_1 \to \tau_2 &= \neg(\tau_1 \times \neg\tau_2) \\
x[\tau_1, \ldots, \tau_n] &= x[\tau_1, \ldots, \tau_n] \\
\forall\alpha : k \,.\, \tau &= \neg(\exists\alpha : k \,.\, \neg\tau) \\
\exists\alpha : k \,.\, \tau &= \exists\alpha : k \,.\, \tau
\end{aligned}
$$

$$
\begin{aligned}
\epsilon &= \epsilon \\
\Gamma, \alpha : k &= \Gamma, \alpha : k \\
\Gamma, x : \tau &= \Gamma, x : \tau \\
[c_1/\alpha]c_2 &= [c_1/\alpha]c_2 \\
\alpha &= \alpha
\end{aligned}
$$

Type directed translation:
Judgement:
$\Gamma \vdash e : \tau \rightsquigarrow x \,.\, e$
Note here that this an expression where we compute the value of e and send it to the continuation x.

Type Principle:
If $\Gamma \vdash e : \tau \rightsquigarrow x \,.\, e$ (and $\vdash \Gamma$ ok) then $\Gamma, k : \neg\tau \vdash e : 0$.

Note: k is not the metavariable for kind in this case.

9.3 $\Gamma \vdash e : \tau \rightsquigarrow k . e$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \rightsquigarrow k : \neg\tau . kx}$$

$$\frac{\Gamma \vdash e : x[\tau_0, \dots, \tau_{n-1} \rightsquigarrow k' : \neg x[\tau_0, \dots, \tau_{n-1}] . e}{\Gamma \vdash \pi_i e : \tau_i \rightsquigarrow k : \not\tau_i . \texttt{let } k' = \lambda x : *[\tau_0, \dots, \tau_{n-1}] . \texttt{let } y = \pi_i x \texttt{ in } ky \texttt{ end in } e \texttt{ end}}$$

$$\frac{\Gamma \vdash e_i : \tau_i \rightsquigarrow k_i : \neg\tau_i . e_i \qquad (i = 1 \dots n)}{\begin{array}{c}\Gamma \vdash \langle e_1, \dots e_n\rangle : *[\tau_i, \dots, \tau_n] \rightsquigarrow \\ k : \neg x[\tau_1, \dots, \tau_n . \texttt{let } k_1 = \lambda x_1 : \tau_1 . \texttt{let } k_2 = \lambda x_2 : \tau_2 . \dots \texttt{let } k_n = \lambda x_n . \tau_n k\langle x_1, \dots, x_n \texttt{ in } e_n \texttt{ end in } e_2 \texttt{ end in } e_1 \texttt{ end}\end{array}}$$

$$\frac{\Gamma \vdash \tau_1 : \text{Type} \qquad \Gamma, x : \tau_1 \vdash e : \tau_2 \rightsquigarrow k' : \not\tau_2 . e}{\Gamma \vdash \lambda x : \tau_1 . e : \tau_1 \to \tau_2 \rightsquigarrow k^{:\not\tau_1 \to \tau_2 = (\not\tau_1 \times \not\tau_2)} . k(\lambda y : \tau_1 \times \tau_2 . \texttt{let } x = \pi_0 y \texttt{ in let } k = \pi_1 y \texttt{ in } e \texttt{ end end})}$$

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \rightsquigarrow k^{:\neg\tau \to \tau' = \neg\neg(\tau_1 \times \neg\tau_2)} . e \qquad \Gamma \vdash e_2 : \tau \rightsquigarrow k_2^{:\neg\tau} . e_2}{\Gamma \vdash e_1\ e_2 : \tau' \rightsquigarrow k^{\neg\tau'} . \texttt{let } k_1 = \lambda f : \neg(\tau \times \neg\tau' . \texttt{let } k_2 = \lambda x : \tau . f\langle x, k\rangle \texttt{ in } e_2 \texttt{ end in } e_1 \texttt{ end}}$$

$$\frac{\Gamma \vdash c : k \qquad \Gamma \vdash e : [c \,/\, \alpha]\tau \rightsquigarrow k' : \neg[c/\alpha]\tau . e \qquad \Gamma, \alpha : k \vdash \tau : \text{Type}}{\Gamma \vdash \texttt{pack}[c, e] \,\text{as}\, \exists\alpha : k . \tau : \exists\alpha : k . e \rightsquigarrow \qquad k^{:\neg\exists\alpha:k.\tau = \neg\exists\alpha:k.\tau} . \texttt{let } k' : \lambda x : [c \,/\, \alpha]\tau . k(\texttt{pack}[c, x] \,\text{as}\, \exists\alpha : k . \tau = \ \texttt{in } e \texttt{ end}}$$

$$\frac{\Gamma \vdash e_1 : \exists\alpha : k . \tau \rightsquigarrow \neg\exists\alpha : k . \tau \qquad \Gamma, \alpha : k, x : \tau \vdash e_2 : \tau' \rightsquigarrow k_2^{:\neg\tau'} . e_2 \qquad \Gamma \vdash \tau : \text{Type}}{\Gamma \vdash \texttt{unpack}[\alpha, x] = e_1 \texttt{in} e_2 \rightsquigarrow \qquad k^{:\neg\tau'} . \texttt{let } k_1 = \lambda x_1 : \exists\alpha : k\tau . \texttt{unpack}[\alpha, x] = x_1 \texttt{in}[k/k_2]e_2 \texttt{ in } e_1 \texttt{ end}}$$

$$\frac{\Gamma \vdash k : \text{kind} \qquad \Gamma, \alpha : k \vdash e : \tau \rightsquigarrow k'^{:\neg\tau} . e}{\Gamma \vdash \Lambda\alpha : k . e : \forall\alpha : k . \tau \rightsquigarrow k^{:\neg\forall\alpha:k.\tau = \neg\neg(\exists\alpha:k.\tau)} . k(\lambda x : \exists\alpha : k . \neg\tau . \texttt{unpack}[\alpha, k'] = x\texttt{in}e)}$$

$$\frac{\Gamma \vdash e : \forall\alpha : k\,.\,\tau \rightsquigarrow k'^{:\neg\forall\alpha:k.\tau = \neg\neg(\exists\alpha.k\neg\tau)}\,.\,e \qquad \Gamma \vdash c : k}{\Gamma \vdash e[c] : [c\,/\,\alpha]\tau \rightsquigarrow k^{:\neg[c/\alpha]\tau}\,.\,\texttt{let}\ k' = \lambda f : \neg(\exists\alpha : k\,.\,\neg\tau)\,.\,f(\mathrm{pack}[c,k]\ \mathrm{as}\ \exists\alpha : k\,.\,\neg\tau)\ \texttt{in}\ e\ \texttt{end}}$$

Note $\neg[c\,/\,\alpha]\tau = \neg[c\,/\,\alpha]\tau$.

9.4 Exceptions

$\tau_1 \to \tau_2 = \neg(\times[\tau_1, \neg\tau_2, \neg\texttt{exn}]$
$\forall\alpha : k\,.\,\tau = \neg(\exists\alpha : k\,.\,\neg\tau \times \neg\texttt{exn})$

Judgement: $\Gamma \vdash e : \tau \rightsquigarrow k^{:\neg\tau} k_{ex}^{:\neg\texttt{exn}}\,.\,e$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \rightsquigarrow k k_{ex}\,.\,kx}$$

$$\frac{\Gamma \vdash e_i : \tau_i \rightsquigarrow k_i^{:\neg\tau_1} k_{ex_i}^{:\neg\texttt{exn}}\,.\,e_i \qquad (i = 1\ldots n)}{\Gamma \vdash \langle e_1, \ldots, e_n\rangle : x[\tau_1, \ldots, \tau_n] \rightsquigarrow k k_{ex}\,.\,\texttt{let}\ k_1 = \lambda x_1 : \tau_1\,.\,\ldots\texttt{let}\ k_n = \lambda x_n : \tau_n\,.\,k\langle x_i, \ldots x_n\rangle\ \texttt{in}\ e_n\ \texttt{end}\ldots\ \texttt{in}\ e_1\ \texttt{end}}$$

$$\frac{\Gamma \vdash e : \mathrm{Type} \qquad \Gamma \vdash e : \texttt{exn} \rightsquigarrow k'^{:\neg\texttt{exn}} k_{ex}'^{:\neg\texttt{exn}}\,.\,e}{\Gamma \vdash \texttt{raise}_\tau e : \tau \rightsquigarrow k^{:\neg\tau} k_{ex}^{:\neg\texttt{exn}}\,.\,\texttt{let}\ k' = k_{ex}\ \texttt{in}\ e\ \texttt{end}}$$

$$\frac{\Gamma \vdash e : \tau \rightsquigarrow k^{:\neg\tau} k_{ex1}\,.\,e_1 \qquad \Gamma, x : \texttt{exn} \vdash e_2 : \tau \rightsquigarrow k^{:\neg\tau} k_{ex}\,.\,e_2}{\Gamma \vdash \texttt{handle}(e_1, x\,.\,e_2 : \tau \rightsquigarrow k^{:\neg\tau} k_{ex}^{:\neg\texttt{exn}}\,.\,\texttt{let}\ k_{em} = \lambda x : \texttt{exn}\,.\,e_2\ \texttt{in}\ e_1\ \texttt{end}}$$

$$\frac{\Gamma \vdash \tau_1 : \mathrm{Type} \qquad \Gamma, x : \tau_1 \vdash e : \tau_2 \rightsquigarrow k'^{\neg\tau_2} k_{ex}'\,.\,e_2}{\begin{array}{c}\Gamma \vdash \lambda x : \tau_1\,.\,e : \tau_1 \to \tau_2 \rightsquigarrow \\ k^{:\neg\tau_1\to\tau_2 = \neg\neg(\times[\tau_1,\neg\tau_2,\neg\texttt{exn}])} k_{ex}\,.\,k(\lambda y : x[\tau_1, \neg\tau_2, \neg\texttt{exn}]\,.\,\texttt{let}\ x = \pi_0 y\ \texttt{in}\ \texttt{let}\ k' = \pi_1 y\ \texttt{in}\ \texttt{let}\ k_{ex}' = \pi_2 y\ \texttt{in}\ e\ \texttt{end}\ \texttt{end}\ \texttt{end})\end{array}}$$

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \rightsquigarrow k_1^{\neg\tau\to\tau' = \neg\neg(\times[\tau,\neg\tau',\neg\texttt{exn}])} k_{ex}\,.\,e_1 \qquad \Gamma \vdash e_2 : \tau \rightsquigarrow k_2^{\neg\tau} k_{ex2}\,.\,e_2}{\Gamma \vdash e_1\ e_2 : \tau' \rightsquigarrow k^{\neg\tau'} k_{ex}\,.\,\texttt{let}\ k_1 = (\lambda f : \neg(x[\tau, \neg\tau', \neg\texttt{exn}])\,.\,\texttt{let}\ k_2 = \lambda x : \tau\,.\,f\langle x, k, k_{ex}\rangle\ \texttt{in}\ e_2\ \texttt{end})\ \texttt{in}\ e_1\ \texttt{end}}$$

9.5 Continuations

```
callcc : ('a cont → 'a) → 'a
throw  : ('a cont × 'a) → 'b
```

Typing Rules

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \qquad \Gamma, x : \texttt{cont}\ \tau \vdash e : \tau}{\Gamma \vdash \texttt{callcc}_\tau x\,.\,e : \tau} \qquad \frac{\Gamma \vdash \tau' : \mathrm{Type} \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \texttt{cont}\ \tau}{\Gamma \vdash \texttt{throw}_{\tau'} e_1\ \texttt{to}\ e_2 : \tau'}$$

Translation Rules

$$\frac{\Gamma \vdash \tau : \mathrm{Type} \qquad \Gamma, x : \texttt{cont}\ \tau \vdash e : \tau \rightsquigarrow k'^{\neg\tau}\,.\,e}{\Gamma \vdash \texttt{callcc}_\tau x\,.\,e : \tau \rightsquigarrow k^{:\neg\tau}\,.\,\texttt{let}\ k' = k\ \texttt{in}\ \texttt{let}\ x = k\ \texttt{in}\ e\ \texttt{end}\ \texttt{end}}$$

$$\frac{\Gamma \vdash \tau' : \mathrm{Type} \qquad \Gamma \vdash e_1 : \tau \rightsquigarrow k_1^{:\neg\tau}\,.\,e_1 \qquad \Gamma \vdash e_2 : \texttt{cont}\ \tau \rightsquigarrow k_2^{:\neg\texttt{cont}\ \tau = \neg\neg\tau}\,.\,e_2}{\Gamma \vdash \texttt{throw}_{\tau'} e_1\ \texttt{to}\ e_2 : \tau' \rightsquigarrow k^{:\neg\tau'}\,.\,\texttt{let}\ k_1 = \lambda x : \tau\,.\,\texttt{let}\ x_2 = \lambda y : \neg\tau\,.\,y\ x\ \texttt{in}\ e_2\ \texttt{end}\ \texttt{in}\ e_1\ \texttt{end}}$$

10 Closure Conversion

A closure is a tuple containing code and the environment that will be passed in as an additional argument to the code. The code is closed, with no open terms.

Concrete example:
$\lambda x : \text{int} \,.\, x + y + z$
$\rightsquigarrow \langle(\lambda x : \text{int} \,.\, \lambda \texttt{env} : \text{int} \times \text{int} \,.\, \texttt{let } y = \pi_0\texttt{env in let } z = \pi_1\texttt{env in } x + y + z \texttt{ end end}), \langle y, z\rangle\rangle$
$e\ 5$
$\rightsquigarrow \texttt{let } f = \pi_0 e \texttt{ in let env} = \pi_1 e \texttt{ in } f\ 5\ \texttt{env end end}$

This can get really messy and really inefficient, so what we really want to do is convert some curried and some tupled functions and turn them into some special internal representation for multi-argument functions.

Types may be different if the environment is different, so when converting, we can try something like, where we don't care about the existential, since we won't be manipulating it:
$\tau_1 \to \tau_2 = \exists \alpha_{\texttt{env}} : \text{Type} \,.\, (\tau_1 \to \alpha_{\texttt{env}} \to \tau_2) \times \alpha_{\texttt{env}}$

10.1 Target Language (IL-Closure)

$\Delta; \Gamma \vdash e : 0$
$\Delta; \Gamma \vdash v : \tau$
$\quad \Gamma ::= \epsilon \mid \Gamma, x : \tau$
$\quad \Delta ::= \epsilon \mid \Delta, \alpha : k$

Only rule additional we need:

$$\frac{\Delta \vdash \tau : \text{Type} \qquad \Gamma; (\epsilon, x : \tau) \vdash e : 0}{\Delta; \Gamma \vdash \lambda x : \tau \,.\, e : \neg\tau}$$

10.2 Type Translation

$$\begin{aligned} \alpha &= \alpha \\ &\vdots \\ \neg\tau &= \exists \alpha_{\texttt{env}} : \text{Type} \,.\, \neg(\tau \times \alpha_{\texttt{env}}) \times \alpha_{\texttt{env}} \\ x[\tau_1, \ldots, \tau_2] &= x[\tau_1, \ldots, \tau_n] \end{aligned}$$

How would we deal with $\forall \alpha : k \,.\, \tau$? Turns out it's really hard. But because of how we got rid of them back during CPS conversion, we don't even have to deal with it anymore!

10.3 Type Principle

If $\Delta; \Gamma \vdash e : 0 \rightsquigarrow e$ then $\Delta; \Gamma \vdash e : 0$.
If $\Delta; \Gamma \vdash v : \tau \rightsquigarrow v$ then $\Delta; \Gamma \vdash v : \tau$.

10.4 $\Delta;\Gamma \vdash e : 0 \rightsquigarrow e$, $\Delta;\Gamma \vdash v : \tau \rightsquigarrow v$

$$\frac{\Gamma(x) = \tau}{\Delta;\Gamma \vdash x : \tau \rightsquigarrow x} \qquad \frac{\Delta;\Gamma \vdash v_i : \tau_i \rightsquigarrow v_i}{\Delta;\Gamma \vdash \langle v_1, \dots v_n\rangle : x[\tau_1, \dots, \tau_n] \rightsquigarrow \langle v_1, \dots, v_n\rangle}$$

$$\frac{\Delta \vdash \tau : \text{Type} \qquad \Delta;\Gamma, x : \tau \vdash e : 0 \rightsquigarrow e \qquad \Gamma = x_1 : \tau_1, \dots, x_n : \tau_n}{\begin{array}{c}\Delta;\Gamma \vdash \lambda x : \tau \,.\, e : \neg\tau \rightsquigarrow \texttt{pack}[x[\tau_1, \dots, \tau_n], \langle \\ (\lambda y : \tau \times x[\tau_1, \dots, \tau_n] \,.\, \texttt{let } x = \pi_0 y \texttt{ in let env} = \pi_1 y \texttt{ in let } x_1 = \pi_0 \texttt{env in } \dots \texttt{let } x_n = \pi_{n-1}\texttt{env in } e \texttt{ end end end end}), \\ \langle x_1, \dots, x_n\rangle\rangle] \text{ as } \exists \alpha_{\texttt{env}} : \text{Type} \,.\, \neg(\tau \times \alpha_{\texttt{env}}) \times \alpha_{\texttt{env}}\end{array}}$$

$$\frac{\Delta;\Gamma \vdash v_1 : \neg\tau \rightsquigarrow v_1{}^{:\neg\tau = \exists\alpha.\neg(\tau\times\alpha)\times\alpha} \qquad \Delta;\Gamma \vdash v_2 : \tau \rightsquigarrow v_2{}^{:\tau}}{\Delta;\Gamma \vdash v_1\ v_2 : 0 \rightsquigarrow \texttt{unpack}[\alpha_{\texttt{env}}, x] = v_1 \texttt{inlet } f = \pi_0 x \texttt{ in let env} = \pi_1 x \texttt{ in } f\langle v_2, \texttt{env}\rangle \texttt{ end end}}$$

To solve some inefficiency, instead of passing around the environment, instead, pass around the closure.
In environment passing, we had $\tau_1 \to \tau_2 = \exists \alpha_{\texttt{env}} : \text{Type} \,.\, (\tau_1 \times \alpha_{\texttt{env}} \to \tau_2) \times \alpha_{\texttt{env}}$.
In closure passing, we have $\mu\beta \,.\, \exists\alpha_{\texttt{env}} \,.\, \tau_1 \times \beta \to \tau_2 \times \alpha_{\texttt{env}}$.

We can apparently use this to understand objected oriented programming better and some of the research might not even be all that wrong.

There's more stuff one can do ....

11 Hoisting

Type Translation:
$\tau = \tau$

11.1 Target Language: IL-Hoist

$$\begin{aligned}
c &::= \cdots \mid \forall\alpha : k \,.\, c \\
e &::= \dots \\
v &::= \cdots \mid \lambda x : \cancel{\tau} \,.\, e \mid v[c] \mid \cancel{\Lambda\alpha : k \,.\, v} \\
f &::= \lambda x : \tau \,.\, e \mid \Lambda\alpha : k \,.\, f \\
b &::= x = f \\
P &::= \texttt{let } b \texttt{ in } P \mid e
\end{aligned}$$

To hoist the type: 'type-erasure semantics'.

11.2 Judgements

$\Delta;\Gamma \vdash e : 0 \rightsquigarrow let\vec{b}in e$
$\Delta;\Gamma \vdash v : \tau \rightsquigarrow let\vec{b}in v$

11.3 $\Delta;\Gamma \vdash f : \tau$

$$\frac{\Delta, \alpha : k;\Gamma \vdash f : \tau}{\Delta;\Gamma \vdash \Lambda\alpha : k \,.\, f : \forall\alpha : k \,.\, \tau} \qquad \frac{\Delta \vdash \tau : \mathrm{Type} \qquad \Delta;\Gamma, x : \tau \vdash e : 0}{\Delta;\Gamma \vdash \lambda x : \tau \,.\, e : \neg\tau}$$

11.4 $\Gamma \vdash P : 0$

$$\frac{\cdot;\Gamma \vdash f : \tau \qquad \Gamma, x : \tau \vdash P : 0}{\Gamma \vdash \texttt{let}\ x = f\ \texttt{in}\ P\ \texttt{end} : 0} \qquad \frac{\cdot;\Gamma \vdash e : 0}{\Gamma \vdash e : 0}$$

11.5 $\Delta;\Gamma \vdash v : \tau \rightsquigarrow \texttt{let}\ \vec{b}\ \texttt{in}\ v\ \texttt{end}$

$$\frac{\Gamma(x) = \tau}{\Delta;\Gamma \vdash x : \tau \rightsquigarrow \texttt{let}\ \ \texttt{in}\ x\ \texttt{end}} \qquad \frac{\Delta;\Gamma \vdash v_i : \tau_i \rightsquigarrow \texttt{let}\ \vec{b_i}\ \texttt{in}\ v_i\ \texttt{end} \qquad (\mathrm{for} i = 1 \ldots n)}{\Delta;\Gamma \vdash \langle v_1 \ldots v_n\rangle : \times[\tau_1 \ldots \tau_n]\texttt{let}\ \vec{b_1} \ldots \vec{b_n}\ \texttt{in}\ \langle\langle v_1 \ldots v_n\rangle\ \texttt{end}}$$

$$\frac{\Delta;\Gamma \vdash v_1 : \neg\tau \rightsquigarrow \texttt{let}\ \vec{b_1}\ \texttt{in}\ v_1\ \texttt{end} \qquad \Delta;\Gamma \vdash v_2 : \tau \rightsquigarrow \texttt{let}\ \vec{b_2}\ \texttt{in}\ v_2\ \texttt{end}}{\Delta;\Gamma \vdash v_1 v_2 : 0 \rightsquigarrow \texttt{let}\ \vec{b_1}, \vec{b_2}\ \texttt{in}\ v_1 v_2\ \texttt{end}}$$

$$\frac{\Delta \vdash \tau : \mathrm{Type} \qquad \Delta; x : \tau \vdash e : 0 \rightsquigarrow \texttt{let}\ \vec{b}\ \texttt{in}\ e\ \texttt{end} \qquad y \notin FV(\Gamma), y \neq x, y \notin BV(\vec{b}) \qquad \Delta = \alpha_1 : k_1, \ldots, \alpha_n : k_n}{\Delta;\Gamma \vdash \lambda x : \tau \,.\, e : \neg\tau \rightsquigarrow \texttt{let}\ \vec{b}, y = \Lambda\alpha_1 : k_1 \ldots . \Lambda\alpha_n : k_n \,.\, \lambda x : \tau \,.\, e\ \texttt{in}\ y[\alpha_1] \ldots [\alpha_n]\ \texttt{end}}$$

11.6 $\Delta;\Gamma \vdash e : 0 \rightsquigarrow \texttt{let}\ \vec{b}\ \texttt{in}\ v\ \texttt{end}$

$$\frac{\cdot;\cdot \vdash e : 0 \rightsquigarrow \texttt{let}\ \vec{b}\ \texttt{in}\ e\ \texttt{end} \qquad (\vec{b} = b_1, \ldots, b_n)}{\vdash_{\texttt{top}} e : 0 \rightsquigarrow \texttt{let}\ b_1\ \texttt{in}\ \ldots \texttt{let}\ b_n\ \texttt{in}\ e\ \texttt{end}\ \texttt{end}}$$

TODO: ASIDE PREVIOUS: CLOSURE CONVERSION

$$\frac{\cdot;\cdot \vdash e : 0 \rightsquigarrow e}{\vdash_{\texttt{top}} e : 0 \rightsquigarrow e}$$

TODO: ASIDE PREVIOUS: CPS CONVERSION

$$\frac{\cdot;\cdot \vdash e : \tau \rightsquigarrow k k_{ex} \,.\, e}{\vdash_{\texttt{top}} e : \tau \rightsquigarrow \atop \texttt{let}\ k = \lambda x : \tau \,.\, \texttt{halt}\ \texttt{in}\ \texttt{let}\ k_{ex} = \lambda x : \texttt{exn} \,.\, \texttt{let}\ \_ = \texttt{print "uncaught exception"}\ \texttt{in}\ \texttt{halt}\ \texttt{end}\ \texttt{in}\ e\ \texttt{end}\ \texttt{end}}$$

12 Alloc

12.1 Target Language: IL-Alloc

$$\begin{aligned} a &::= x = \texttt{alloc}[n] \mid x = \pi_i y \mid \pi_i y \mathrel{::-} v \mid x = v \\ e &::= \texttt{let}\ a\ \texttt{in}\ e\ \texttt{end} \mid x\ x \mid \texttt{halt} \\ v &::= x \mid \cdots \mid \cancel{\langle v_1 \ldots v_n \rangle} \\ f &::= \lambda x : \tau\ .\ e \\ b &::= x = f \\ P &::= \texttt{let}\ b\ \texttt{in}\ P \mid e \end{aligned}$$

12.2 Judgements and Translations

$\Delta; \Gamma \vdash e : 0 \rightsquigarrow e$
$\Delta; \Gamma \vdash v : \tau \rightsquigarrow \texttt{let}\ \vec{a}\ \texttt{in}\ v\ \texttt{end}$

12.3 $\Delta; \Gamma \vdash v : \tau \rightsquigarrow \texttt{let}\ \vec{a}\ \texttt{in}\ v\ \texttt{end}$

$$\frac{\Delta; \Gamma \vdash v_i : \tau_i \rightsquigarrow \texttt{let}\ \vec{a_i}\ \texttt{in}\ v_i\ \texttt{end} \qquad (\texttt{for}\ i = 1 \ldots n) \qquad \texttt{fresh}[x]}{\Delta; \Gamma \vdash \langle v_1 \ldots v_n \rangle : \times[\tau_1 \ldots \tau_n] \rightsquigarrow \texttt{let}\ \vec{a_1} \ldots \vec{a_n}, x = \texttt{alloc}[n], \pi_0 x = v_1 \ldots, \pi_{n-1} x = v_n\ \texttt{in}\ x\ \texttt{end}}$$

12.4 $\Delta; \Gamma \vdash e : 0 \rightsquigarrow e$

$$\frac{\Delta; \Gamma \vdash v : \tau \rightsquigarrow \texttt{let}\ \vec{a}\ \texttt{in}\ i\ \texttt{end} nv \qquad \Delta; \Gamma, x : \tau \vdash e : 0 \rightsquigarrow e \qquad \vec{a} = a_1, \ldots, a_n}{\Delta; \Gamma \vdash \texttt{let}\ x = v\ \texttt{in}\ e\ \texttt{end} : 0 \rightsquigarrow \texttt{let}\ a_1\ \texttt{in}\ \ldots \texttt{let}\ a_n\ \texttt{in}\ \texttt{let}\ x = v\ \texttt{in}\ e\ \texttt{end}\ \texttt{end}\ \texttt{end}}$$

$$\frac{\Delta; \Gamma \vdash v_1 : \neg\tau \rightsquigarrow \texttt{let}\ \vec{a_1}\ \texttt{in}\ v_1\ \texttt{end} \qquad \Delta; \Gamma \vdash v_2 : \tau \rightsquigarrow \texttt{let}\ \vec{a_2}\ \texttt{in}\ v_2\ \texttt{end}}{\Delta; \Gamma \vdash v_1 v_2 : 0 \rightsquigarrow \texttt{let}\ \vec{a_1}\ \texttt{in}\ \texttt{let}\ \vec{a_2}\ \texttt{in}\ v_1 v_2\ \texttt{end}\ \texttt{end}}$$

13 Module

13.1 Target Language: IL-Module

$$\begin{aligned} k &::= \text{Type} \mid S(c) \mid \Pi\alpha : k_1\ .\ k_2 \mid \Sigma\alpha : k_1\ .\ k_2 \mid 1 \\ c &::= \cdots \mid c_1 \to c_2 \mid \forall\alpha : k\ .\ c \mid \ldots \\ e &::= \cdots \mid \texttt{ext}\ M \\ \sigma &::= 1 \mid (\!| k |\!) \mid \langle\!| \tau |\!\rangle \mid \Pi^{\mathsf{gen}}\alpha : \sigma_1\ .\ \sigma_2 \mid \Pi^{\mathsf{app}}\alpha : \sigma_1\ .\ \sigma_2 \mid \Sigma\alpha : \sigma_1\ .\ \sigma_2 \\ M &::= * \mid (\!| k |\!) \mid \langle\!| e |\!\rangle \end{aligned}$$