Symbol Classes

$$\begin{array}{lcl}
a, b, c, co & \rightarrow & \langle\text{type variable}\rangle \\
x, f & \rightarrow & \langle\text{term variable}\rangle \\
C & \rightarrow & \langle\text{coercion constant}\rangle \\
T & \rightarrow & \langle\text{value type constructor}\rangle \\
S_n & \rightarrow & \langle n\text{-ary type function}\rangle \\
K & \rightarrow & \langle\text{data constructor}\rangle
\end{array}$$

Declarations

$$\begin{array}{lcl}
pgm & \rightarrow & \overline{decl};\ e \\
decl & \rightarrow & \mathbf{data}\, T : \overline{\kappa} \rightarrow \star\ \mathbf{where} \\
 & & \quad \overline{K : \forall \overline{a:\kappa}.\, \forall \overline{b:\iota}.\ \overline{\sigma} \rightarrow T\ \overline{a}} \\
 & \mid & \mathbf{type}\, S_n : \overline{\kappa}^n \rightarrow \iota \\
 & \mid & \mathbf{axiom}\, C : \sigma_1 \sim \sigma_2
\end{array}$$

Sorts and kinds

$$\begin{array}{lcll}
\delta & \rightarrow & \textsc{ty} \mid \textsc{co} & \text{Sorts} \\
\kappa, \iota & \rightarrow & \star \mid \kappa_1 \rightarrow \kappa_2 \mid \sigma_1 \sim \sigma_2 & \text{Kinds}
\end{array}$$

Types and Coercions

$$\begin{array}{lcl}
d & \rightarrow & a \mid T \qquad \text{Atom of sort } \textsc{ty} \\
g & \rightarrow & c \mid C \qquad \text{Atom of sort } \textsc{co} \\
\varphi, \rho, \sigma, \tau, \upsilon, \gamma & \rightarrow & a \mid C \mid T \mid \varphi_1\ \varphi_2 \mid S_n\ \overline{\varphi}^n \mid \forall a:\kappa.\, \varphi \\
 & \mid & \mathrm{sym}\,\gamma \mid \gamma_1 \circ \gamma_2 \mid \gamma @ \varphi \mid \mathrm{left}\,\gamma \mid \mathrm{right}\,\gamma \\
 & \mid & \gamma \sim \gamma \mid \mathrm{rightc}\,\gamma \mid \mathrm{leftc}\,\gamma \mid \gamma \blacktriangleright \gamma
\end{array}$$

We use ρ, σ, τ, and υ for regular types, γ for coercions, and φ for both.

Syntactic sugar

$$\text{Types} \quad \kappa \Rightarrow \sigma \quad \equiv \quad \forall\_ : \kappa.\, \sigma$$

Terms

$$\begin{array}{lcll}
u & \rightarrow & x \mid K & \text{Variables and data constructors} \\
e & \rightarrow & u & \text{Term atoms} \\
 & \mid & \Lambda a:\kappa.\, e \mid e\ \varphi & \text{Type abstraction/application} \\
 & \mid & \lambda x:\sigma.\, e \mid e_1\ e_2 & \text{Term abstraction/application} \\
 & \mid & \mathbf{let}\ x:\sigma = e_1\ \mathbf{in}\ e_2 & \\
 & \mid & \mathbf{case}\ e_1\ \mathbf{of}\ \overline{p \rightarrow e_2} & \\
 & \mid & e \blacktriangleright \gamma & \text{Cast} \\
 & & & \\
p & \rightarrow & K\ \overline{b:\kappa}\ \overline{x:\sigma} & \text{Pattern}
\end{array}$$

Environments

$$\Gamma \quad \rightarrow \quad \epsilon \mid \Gamma, u:\sigma \mid \Gamma, d:\kappa \mid \Gamma, g:\kappa \mid \Gamma, S_n:\kappa$$

A *top-level environment* binds only type constructors, T, S_n, data constructors K, and coercion constants C.