$$
\begin{array}{rcll}
\textsf{BinOp} & \oplus & ::= & \texttt{Product} \mid \texttt{Sum} \mid \texttt{Arrow} \\
\textsf{Kind} & \kappa & ::= & \texttt{Ty} \mid \texttt{KHole} \mid \texttt{S(}\tau\texttt{)} \\
\textsf{ConstantTypes} & c & ::= & \texttt{Int} \mid \texttt{Float} \mid \texttt{Bool} \\
\textsf{UserHTyp} & \hat{\tau} & ::= & c \mid \hat{\tau}_1 \oplus \hat{\tau}_2 \mid \texttt{list(}\hat{\tau}\texttt{)} \mid (\!|\,|\!)^u \mid (\!|\hat{\tau}|\!)^u \\
\textsf{InternalHTyp} & \tau & ::= & c \mid \tau_1 \oplus \tau_2 \mid \texttt{list(}\tau\texttt{)} \mid (\!|\,|\!)^u \mid (\!|\tau|\!)^u \\
\textsf{TypeVars} & t & & \\
\textsf{TypePattern} & \rho & ::= & t \mid (\!|\,|\!) \mid (\!|t|\!) \\
\textsf{UserExpression} & e & ::= & \texttt{type}\ \rho\ \texttt{=}\ \hat{\tau}\ \texttt{in}\ e \mid \textit{elided} \\
\textsf{InternalExpression} & \tau & ::= & \texttt{type}\ \rho\ \texttt{=}\ \tau : \kappa\ \texttt{in}\ d \mid \textit{elided}
\end{array}
$$

$\boxed{\Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2}$ κ_1 is a consistent subkind of κ_2

$$
\texttt{KCHoleL}\ \frac{}{\Delta; \Phi \vdash \texttt{KHole} \lesssim \kappa}
\qquad
\texttt{KCHoleR}\ \frac{}{\Delta; \Phi \vdash \kappa \lesssim \texttt{KHole}}
\qquad
\texttt{KCRespectEquiv}\ \frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2}
$$

$$
\texttt{KCSubsumption}\ \frac{\Delta; \Phi \vdash \tau \Leftarrow \texttt{Ty}}{\Delta; \Phi \vdash \texttt{S(}\tau\texttt{)} \lesssim \texttt{Ty}}
$$

$\boxed{t\ \texttt{valid}}$ t is a valid type variable

t is valid if it is not a builtin-type or keyword, begins with an alpha char or underscore, and only contains alphanumeric characters, underscores, and primes.

$\boxed{\Delta; \Phi \vdash \kappa\ \texttt{kind}}$ κ forms a kind

$$
\texttt{KFTy}\ \frac{}{\Delta; \Phi \vdash \texttt{Ty}\ \texttt{kind}}
\qquad
\texttt{KFHole}\ \frac{}{\Delta; \Phi \vdash \texttt{KHole}\ \texttt{kind}}
\qquad
\texttt{KFSing}\ \frac{\Delta; \Phi \vdash \tau \Leftarrow \texttt{Ty}}{\Delta; \Phi \vdash \texttt{S(}\tau\texttt{)}\ \texttt{kind}}
$$

$\boxed{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}$ κ_1 is equivalent to κ_2

$$\texttt{KERefl}\ \frac{}{\Delta; \Phi \vdash \kappa \equiv \kappa} \qquad \texttt{KESymm}\ \frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_2 \equiv \kappa_1} \qquad \texttt{KETrans}\ \frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2 \qquad \Delta; \Phi \vdash \kappa_2 \equiv \kappa_3}{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_3}$$

$$\texttt{KESingEquiv}\ \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash \texttt{S}(\tau_1) \equiv \texttt{S}(\tau_2)}$$

$\boxed{\Delta; \Phi \vdash \tau \Rightarrow \kappa}$ τ synthesizes kind κ

$$\texttt{KSConst}\ \frac{}{\Delta; \Phi \vdash c \Rightarrow \texttt{S}(c)} \qquad \texttt{KSVar}\ \frac{t : \kappa \in \Phi}{\Delta; \Phi \vdash t \Rightarrow \kappa} \qquad \texttt{KSUVar}\ \frac{t \notin \mathsf{dom}(\Phi)}{\Delta; \Phi \vdash t \Rightarrow \texttt{KHole}}$$

$$\texttt{KSBinOp}\ \frac{\Delta; \Phi \vdash \tau_1 \Leftarrow \texttt{S}(\tau_1) \qquad \Delta; \Phi \vdash \tau_2 \Leftarrow \texttt{S}(\tau_2)}{\Delta; \Phi \vdash \tau_1 \oplus \tau_2 \Rightarrow \texttt{S}(\tau_1 \oplus \tau_2)}$$

$$\texttt{KSList}\ \frac{\Delta; \Phi \vdash \tau \Leftarrow \texttt{S}(\tau)}{\Delta; \Phi \vdash \texttt{list}(\tau) \Rightarrow \texttt{S}(\texttt{list}(\tau))} \qquad \texttt{KSEHole}\ \frac{u :: \kappa \in \Delta}{\Delta; \Phi \vdash (\!|\,|\!)^u \Rightarrow \kappa}$$

$$\texttt{KSNEHole}\ \frac{u :: \kappa \in \Delta \qquad \Delta; \Phi \vdash \tau \Rightarrow \kappa'}{\Delta; \Phi \vdash (\!|\tau|\!)^u \Rightarrow \kappa}$$

$\boxed{\Delta; \Phi \vdash \tau \Leftarrow \kappa}$ τ analyzes against kind κ

$$\texttt{KAASubsume}\ \frac{\Phi \vdash \tau \Rightarrow \kappa' \qquad \Delta; \Phi \vdash \kappa' \lesssim \kappa}{\Delta; \Phi \vdash \tau \Leftarrow \kappa}$$

$\boxed{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}$ τ_1 is equivalent to τ_2

$$\texttt{KCESymm}\ \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash \tau_2 \equiv \tau_1}$$

$$\texttt{KCETrans}\ \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 \qquad \Delta; \Phi \vdash \tau_2 \equiv \tau_3}{\Delta; \Phi \vdash \tau_1 \equiv \tau_3}$$

$$\texttt{KCESingEquiv}\ \frac{\Delta; \Phi \vdash \tau_1 \Leftarrow \texttt{S}(\tau_2)}{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}$$

$$\texttt{KCEConst}\ \frac{}{\Delta; \Phi \vdash c \equiv c}$$

$$\texttt{KCEVar}\ \frac{t : \kappa \in \Phi}{\Delta; \Phi \vdash t \equiv t}$$

$$\texttt{KCEBinOp}\ \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 \qquad \Delta; \Phi \vdash \tau_3 \equiv \tau_4}{\Delta; \Phi \vdash \tau_1 \oplus \tau_3 \equiv \tau_2 \oplus \tau_4}$$

$$\texttt{KCEList}\ \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash \texttt{list}(\tau_1) \equiv \texttt{list}(\tau_2)}$$

$$\texttt{KCEEHole}\ \frac{u :: \kappa \in \Delta}{\Delta; \Phi \vdash (\!|\!)^u \equiv (\!|\!)^u}$$

$$\texttt{KCENEHole}\ \frac{u :: \kappa \in \Delta \qquad \Delta; \Phi \vdash \tau \Leftarrow \kappa'}{\Delta; \Phi \vdash (\!|\tau|\!)^u \equiv (\!|\tau|\!)^u}$$

$\boxed{\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta}$ $\hat{\tau}$ synthesizes kind κ and elaborates to τ

$$\texttt{TElabSConst}\quad \frac{}{\Phi \vdash c \Rightarrow \texttt{S}(c) \rightsquigarrow c \dashv \cdot}$$

$$\texttt{TElabSBinOp}\quad \frac{\Phi \vdash \hat{\tau}_1 \Leftarrow \texttt{Ty} \rightsquigarrow \tau_1 \dashv \Delta_1 \qquad \Phi \vdash \hat{\tau}_2 \Leftarrow \texttt{Ty} \rightsquigarrow \tau_2 \dashv \Delta_2}{\Phi \vdash \hat{\tau}_1 \oplus \hat{\tau}_2 \Rightarrow \texttt{S}(\tau_1 \oplus \tau_2) \rightsquigarrow \tau_1 \oplus \tau_2 \dashv \Delta_1 \cup \Delta_2}$$

$$\texttt{TElabSList}\quad \frac{\Phi \vdash \hat{\tau} \Leftarrow \texttt{Ty} \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash \texttt{list}(\hat{\tau}) \Rightarrow \texttt{S}(\texttt{list}(\tau)) \rightsquigarrow \texttt{list}(\tau) \dashv \Delta}$$

$$\texttt{TElabSVar}\quad \frac{t : \kappa \in \Phi}{\Phi \vdash t \Rightarrow \texttt{S}(t) \rightsquigarrow t \dashv \cdot}$$

$$\texttt{TElabSUVar}\quad \frac{t \notin \mathsf{dom}(\Phi)}{\Phi \vdash t \Rightarrow \texttt{KHole} \rightsquigarrow \llparenthesis t \rrparenthesis^u \dashv u :: \texttt{KHole}}$$

$$\texttt{TElabSHole}\quad \frac{}{\Phi \vdash \llparenthesis\rrparenthesis^u \Rightarrow \texttt{KHole} \rightsquigarrow \llparenthesis\rrparenthesis^u \dashv u :: \texttt{KHole}}$$

$$\texttt{TElabSNEHole}\quad \frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash \llparenthesis \hat{\tau} \rrparenthesis^u \Rightarrow \texttt{KHole} \rightsquigarrow \llparenthesis \tau \rrparenthesis^u \dashv \Delta, u :: \texttt{KHole}}$$

$\boxed{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$ $\hat{\tau}$ analyzes against kind κ_1 and elaborates to τ

$$\texttt{TElabASubsume}\quad \frac{\hat{\tau} \neq \llparenthesis\rrparenthesis^u \qquad \hat{\tau} \neq \llparenthesis \hat{\tau}' \rrparenthesis^u \qquad \Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta \qquad \Delta; \Phi \vdash \kappa' \lesssim \kappa}{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$$

$$\texttt{TElabAEHole}\quad \frac{}{\Phi \vdash \llparenthesis\rrparenthesis^u \Leftarrow \kappa \rightsquigarrow \llparenthesis\rrparenthesis^u \dashv u :: \kappa}$$

$$\texttt{TElabANEHole}\quad \frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash \llparenthesis \hat{\tau} \rrparenthesis^u \Leftarrow \kappa \rightsquigarrow \llparenthesis \tau \rrparenthesis^u \dashv \Delta, u :: \kappa}$$

$\boxed{\Phi_1 \vdash \tau : \kappa \rhd \rho \dashv \Phi_2}$ ρ matches against $\tau : \kappa$ extending Φ if necessary

$$\texttt{RESVar}\quad \frac{t \ \texttt{valid}}{\Phi \vdash \tau : \kappa \rhd t \dashv \Phi, t :: \kappa}$$

$$\texttt{RESEHole}\quad \frac{}{\Phi \vdash \tau : \kappa \rhd \llparenthesis\rrparenthesis \dashv \Phi}$$

$$\texttt{RESVarHole}\quad \frac{\neg(t \ \texttt{valid})}{\Phi \vdash \tau : \kappa \rhd \llparenthesis t \rrparenthesis \dashv \Phi}$$

$\boxed{\Gamma; \Phi \vdash e \Rightarrow \tau \rightsquigarrow d \dashv \Delta}$ e synthesizes type τ and elaborates to d

ESDefine

$$\frac{\Phi_1 \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta_1 \qquad \Phi_1 \vdash \tau : \kappa \rhd \rho \dashv \Phi_2 \qquad \Gamma; \Phi_2 \vdash e \Rightarrow \tau_1 \rightsquigarrow d \dashv \Delta_2}{\Gamma; \Phi_1 \vdash \texttt{type}\ \rho\ \texttt{=}\ \hat{\tau}\ \texttt{in}\ e \Rightarrow \tau_1 \rightsquigarrow \texttt{type}\ \rho\ \texttt{=}\ \tau : \kappa\ \texttt{in}\ d \dashv \Delta_1 \cup \Delta_2}$$

$\boxed{\Delta; \Gamma; \Phi \vdash d : \tau}$ d is assigned type τ

DEDefine

$$\frac{\Phi_1 \vdash \tau_1 : \kappa \rhd \rho \dashv \Phi_2 \qquad \Delta; \Gamma; \Phi_2 \vdash d : \tau_2}{\Delta; \Gamma; \Phi_1 \vdash \texttt{type}\ \rho\ \texttt{=}\ \tau_1 : \kappa\ \texttt{in}\ d : \tau_2}$$

Theorem 1 (Well-Kinded Elaboration)
(1) If $\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta$ *then* $\Delta; \Phi \vdash \tau \Rightarrow \kappa$
(2) If $\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta$ *then* $\Delta; \Phi \vdash \tau \Leftarrow \kappa$

This is like the Typed Elaboration theorem in the POPL19 paper.

Theorem 2 (Elaborability)
(1) $\exists \Delta$ *s.t. if* $\Delta; \Phi \vdash \tau \Rightarrow \kappa$ *then* $\exists \hat{\tau}$ *such that* $\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta$
(2) $\exists \Delta$ *s.t. if* $\Delta; \Phi \vdash \tau \Leftarrow \kappa$ *then* $\exists \hat{\tau}$ *such that* $\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta$

This is similar but a little different from Elaborability theorem in the POPL19 paper. Choose the Δ that is emitted from elaboration and then there's an $\hat{\tau}$ that elaborates to any of the τ forms. Elaborability and Well-Kinded Elaboration implies we can just rely on the elaboration forms for the premises of any rules that demand kind synthesis/analysis.

Theorem 3 (Type Elaboration Unicity)
(1) If $\Phi \vdash \hat{\tau} \Rightarrow \kappa_1 \rightsquigarrow \tau_1 \dashv \Delta_1$ *and* $\Phi \vdash \hat{\tau} \Rightarrow \kappa_2 \rightsquigarrow \tau_2 \dashv \Delta_2$ *then* $\kappa_1 = \kappa_2$, $\tau_1 = \tau_2$, $\Delta_1 = \Delta_2$
(2) If $\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau_1 \dashv \Delta_1$ *and* $\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau_2 \dashv \Delta_2$ *then* $\tau_1 = \tau_2$, $\Delta_1 = \Delta_2$

This is like the Elaboration Unicity theorem in the POPL19 paper.

Theorem 4 (Kind Synthesis Precision)
If $\Delta; \Phi \vdash \tau \Rightarrow \kappa_1$ *and* $\Delta; \Phi \vdash \tau \Leftarrow \kappa_2$ *then* $\Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2$

Kind Synthesis Precision says that synthesis finds the most precise kappa possible for a given input type. This is somewhat trivial, but interesting to note because it means we can expect singletons wherever possible.