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

$\boxed{\Delta; \Phi \vdash \kappa_1 \leq \kappa_2}$ κ_1 is more precise than κ_2

$$\texttt{KLTrans}\quad \frac{\Delta; \Phi \vdash \kappa_1 \leq \kappa_2 \qquad \Delta; \Phi \vdash \kappa_2 \leq \kappa_3}{\Delta; \Phi \vdash \kappa_1 \leq \kappa_3}$$

$$\texttt{KLTyHole}\quad \frac{}{\Delta; \Phi \vdash \texttt{Ty} \leq \texttt{KHole}}$$

$$\texttt{KLSingletonTy}\quad \frac{\Delta; \Phi \vdash \tau : \texttt{Ty}}{\Delta; \Phi \vdash \texttt{S}(\tau) \leq \texttt{Ty}}$$

$$\texttt{KLRespectEquiv}\quad \frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_1 \leq \kappa_2}$$

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

$$\texttt{KCHoleL}\quad \frac{}{\Delta; \Phi \vdash \texttt{KHole} \lesssim \kappa}$$

$$\texttt{KCHoleR}\quad \frac{}{\Delta; \Phi \vdash \kappa \lesssim \texttt{KHole}}$$

$$\texttt{KCRespectEquiv}\quad \frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2}$$

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

$\boxed{t\ \texttt{valid}}$ 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.

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

KFTy
$$\frac{}{\Delta; \Phi \vdash \mathtt{Ty}\ \mathtt{kind}}$$

KFHole
$$\frac{}{\Delta; \Phi \vdash \mathtt{KHole}\ \mathtt{kind}}$$

KFSing
$$\frac{\Delta; \Phi \vdash \tau : \mathtt{Ty}}{\Delta; \Phi \vdash \mathtt{S}(\tau)\ \mathtt{kind}}$$

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

KERefl
$$\frac{}{\Delta; \Phi \vdash \kappa \equiv \kappa}$$

KESymm
$$\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_2 \equiv \kappa_1}$$

KETrans
$$\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2 \qquad \Delta; \Phi \vdash \kappa_2 \equiv \kappa_3}{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_3}$$

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

$\boxed{\Delta; \Phi \vdash \tau : \kappa}$ τ is assigned kind κ

KAConst
$$\frac{}{\Delta; \Phi \vdash c : \mathtt{Ty}}$$

KAVar
$$\frac{t : \kappa_1 \in \Phi}{\Delta; \Phi \vdash t : \kappa_2}$$

KABinOp
$$\frac{\Delta; \Phi \vdash \tau_1 : \mathtt{Ty} \qquad \Delta; \Phi \vdash \tau_2 : \mathtt{Ty}}{\Delta; \Phi \vdash \tau_1 \oplus \tau_2 : \mathtt{Ty}}$$

KAList
$$\frac{\Delta; \Phi \vdash \tau : \mathtt{Ty}}{\Delta; \Phi \vdash \mathtt{list}(\tau) : \mathtt{Ty}}$$

KAEHole
$$\frac{u :: \kappa[\Phi'] \in \Delta \qquad \Delta; \Phi \vdash \sigma : \Phi'}{\Delta; \Phi \vdash \llparenthesis\rrparenthesis^u_\sigma : \kappa}$$

KANEHole
$$\frac{\Delta; \Phi \vdash \tau : \kappa' \qquad u :: \kappa[\Phi'] \in \Delta \qquad \Delta; \Phi \vdash \sigma : \Phi'}{\Delta; \Phi \vdash \llparenthesis\tau\rrparenthesis^u_\sigma : \kappa}$$

KASelfRecognition
$$\frac{\Delta; \Phi \vdash \tau : \mathtt{Ty}}{\Delta; \Phi \vdash \tau : \mathtt{S}(\tau)}$$

KASubkind
$$\frac{\Delta; \Phi \vdash \tau : \kappa_1 \qquad \Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2}{\Delta; \Phi \vdash \tau : \kappa_2}$$

$\boxed{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 : \kappa}$ τ_1 is equivalent to τ_2 and has kind κ_2

$$\texttt{KCERefl}\quad \frac{\Delta; \Phi \vdash \tau : \kappa}{\Delta; \Phi \vdash \tau \equiv \tau : \kappa} \qquad \texttt{KCESymm}\quad \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 : \kappa}{\Delta; \Phi \vdash \tau_2 \equiv \tau_1 : \kappa}$$

$$\texttt{KCETrans}\quad \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 : \kappa \qquad \Delta; \Phi \vdash \tau_2 \equiv \tau_3 : \kappa}{\Delta; \Phi \vdash \tau_1 \equiv \tau_3 : \kappa} \qquad \texttt{KCESingEquiv}\quad \frac{\Delta; \Phi \vdash \tau_1 : \texttt{S}(\tau_2)}{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 : \texttt{Ty}}$$

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

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

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

$$\texttt{TElabSList}\quad \frac{\Phi \vdash \hat{\tau} \Leftarrow \texttt{Ty} \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash \texttt{list}(\hat{\tau}) \Rightarrow \texttt{S}(\texttt{list}(\tau)) \rightsquigarrow \texttt{list}(\tau) \dashv \Delta} \qquad \texttt{TElabSVar}\quad \frac{t : \kappa \in \Phi}{\Phi \vdash t \Rightarrow \kappa \rightsquigarrow t \dashv \cdot}$$

$$\texttt{TElabSUVar}\quad \frac{t \notin \Phi}{\Phi \vdash t \Rightarrow \texttt{KHole} \rightsquigarrow (\!|t|\!)^u_{\mathsf{id}(\Phi)} \dashv u :: (\!|\,|\!)[\Phi]}$$

$$\texttt{TElabSHole}\quad \frac{}{\Phi \vdash (\!|\,|\!)^u \Rightarrow \texttt{KHole} \rightsquigarrow (\!|\,|\!)^u_{\mathsf{id}(\Phi)} \dashv u :: (\!|\,|\!)[\Phi]}$$

$$\texttt{TElabSNEHole}\quad \frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash (\!|\hat{\tau}|\!)^u \Rightarrow \texttt{KHole} \rightsquigarrow (\!|\tau|\!)^u_{\mathsf{id}(\Phi)} \dashv \Delta, u :: (\!|\,|\!)[\Phi]}$$

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

TElabASubsume

$$\frac{\hat{\tau} \neq t \text{ where } t \notin \Phi \quad \hat{\tau} \neq (\!|\,|\!)^u \quad \hat{\tau} \neq (\!|\hat{\tau}'|\!)^u \quad \Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta \quad \Delta; \Phi \vdash \kappa' \lesssim \kappa}{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$$

TElabAUVar

$$\frac{t \notin \Phi}{\Phi \vdash t \Leftarrow \texttt{KHole} \rightsquigarrow (\!|t|\!)^u_{\mathsf{id}(\Phi)} \dashv u :: (\!|\,|\!)[\Phi]}$$

TElabAEHole

$$\frac{}{\Phi \vdash (\!|\,|\!)^u \Leftarrow \kappa \rightsquigarrow (\!|\,|\!)^u_{\mathsf{id}(\Phi)} \dashv u :: \kappa[\Phi]}$$

TElabANEHole

$$\frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash (\!|\hat{\tau}|\!)^u \Leftarrow \kappa \rightsquigarrow (\!|\tau|\!)^u_{\mathsf{id}(\Phi)} \dashv \Delta, u :: \kappa[\Phi]}$$

$\boxed{\Delta_1; \Phi_1 \vdash \tau : \kappa \rhd \rho \dashv \Phi_2; \Delta_2}$ ρ matches against $\tau : \kappa$ extending the relevant contexts

RESVar

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

RESEHole

$$\frac{}{\Delta; \Phi \vdash \tau : \kappa \rhd (\!|\,|\!)^u \dashv \Phi; \Delta, u :: (\!|\,|\!)[\Phi]}$$

RESVarHole

$$\frac{\neg(t \;\; \texttt{valid})}{\Delta; \Phi \vdash \tau : \kappa \rhd (\!|t|\!)^u \dashv \Phi; \Delta, u :: (\!|\,|\!)[\Phi]}$$

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

ESDefine

$$\frac{\Phi_1 \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta_1 \quad \Delta_1; \Phi_1 \vdash \tau : \kappa \rhd \rho \dashv \Phi_2; \Delta_2 \quad \Gamma; \Phi_2 \vdash e \Rightarrow \tau_1 \rightsquigarrow d \dashv \Delta_3}{\Gamma; \Phi_1 \vdash \texttt{type}\ \rho\ \texttt{=}\ \hat{\tau}\ \texttt{in}\ e \Rightarrow \tau_1 \rightsquigarrow \texttt{type}\ \rho\ \texttt{=}\ \tau : \kappa\ \texttt{in}\ d \dashv \Delta_2 \cup \Delta_3}$$

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

DEDefine

$$\frac{\Delta_1; \Phi_1 \vdash \tau_1 : \kappa \rhd \rho \dashv \Phi_2; \Delta_2 \quad \Delta_2; \Gamma; \Phi_2 \vdash d : \tau_2}{\Delta; \Gamma; \Phi_1 \vdash \texttt{type}\ \rho\ \texttt{=}\ \tau_1 : \kappa\ \texttt{in}\ d : \tau_2}$$

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

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

Theorem 2 (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 3 (Kind Synthesis Precision)
If $\Phi \vdash \hat{\tau} \Rightarrow \kappa_1 \rightsquigarrow \tau \dashv \Delta_1$ *and* $\Delta; \Phi \vdash \tau : \kappa_2$ *then* $\Delta; \Phi \vdash \kappa_1 \leq \kappa_2$

Kind Synthesis Precision says that elaboration synthesizes the most precise kappa possible for a given input type. The proof goes by induction on the elaboration rules and then for each tau, induction on all valid kind assignments for that tau ensuring that each one assignment is greater in the lattice than the kappa synthesized by elaboration.