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

TODO: 1. think about decidability of analysis, and equivalence

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\texttt{KAVar}\quad \frac{t : \kappa \in \Phi}{\Delta; \Phi \vdash t : \kappa}$$

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

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

$$\texttt{KAEHole}\quad \frac{u :: \kappa[\Phi'] \in \Delta \qquad \Delta; \Phi \vdash \sigma : \Phi'}{\Delta; \Phi \vdash (\!|\!)^{u}_{\sigma} : \kappa}$$

$$\texttt{KANEHole}\quad \frac{\Delta; \Phi \vdash \tau : \kappa' \qquad u :: \kappa[\Phi'] \in \Delta \qquad \Delta; \Phi \vdash \sigma : \Phi'}{\Delta; \Phi \vdash (\!|\tau|\!)^{u}_{\sigma} : \kappa}$$

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

$$\texttt{KASubsumption}\quad \frac{\Delta; \Phi \vdash \tau : \kappa_1 \qquad \Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2}{\Delta; \Phi \vdash \tau : \kappa_2}$$

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

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

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

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

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

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

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

$$\texttt{KAList}\quad \frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 : \texttt{Ty}}{\Delta; \Phi \vdash \texttt{list}(\tau_1) \equiv \texttt{list}(\tau_2) : \texttt{Ty}}$$

$$\texttt{KAEHole}\quad \frac{\begin{array}{c} u_1 :: \kappa_1[\Phi_1'] \in \Delta \\ \Delta; \Phi \vdash \sigma_2 : \Phi_1' \qquad u_2 :: \kappa_2[\Phi_2'] \in \Delta \qquad \Delta; \Phi \vdash \sigma_2 : \Phi_2' \qquad \Delta; \Phi \vdash \kappa_1 \equiv \kappa_2 \end{array}}{\Delta; \Phi \vdash (\!|\!)^{u_1}_{\sigma_1} \equiv (\!|\!)^{u_2}_{\sigma_2} : \kappa}$$

$$\texttt{KANEHole}\quad \frac{\begin{array}{c} \Delta; \Phi \vdash \tau_1 : \kappa_1' \qquad \Delta; \Phi \vdash \tau_2 : \kappa_2' \qquad u_1 :: \kappa_1[\Phi_1'] \in \Delta \\ \Delta; \Phi \vdash \sigma_2 : \Phi_1' \qquad u_2 :: \kappa_2[\Phi_2'] \in \Delta \qquad \Delta; \Phi \vdash \sigma_2 : \Phi_2' \qquad \Delta; \Phi \vdash \kappa_1 \equiv \kappa_2 \end{array}}{\Delta; \Phi \vdash (\!|\tau_1|\!)^{u_1}_{\sigma_1} \equiv (\!|\tau_2|\!)^{u_2}_{\sigma_2} : \kappa}$$

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

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

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

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

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

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

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

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

$$\texttt{TElabASubsume}\quad \frac{\begin{array}{c}\hat{\tau} \neq t \text{ where } t \notin \mathsf{dom}(\Phi) \\ \hat{\tau} \neq (\!|\,|\!)^u \qquad \hat{\tau} \neq (\!|\hat{\tau}'|\!)^u \qquad \Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta \qquad \Delta; \Phi \vdash \kappa' \lesssim \kappa\end{array}}{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$$

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

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

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

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

$$\texttt{RESVar}\quad \frac{t\ \texttt{valid}}{\Phi \vdash \tau : \kappa \rhd t \dashv \Phi, t :: \kappa} \qquad \texttt{RESEHole}\quad \frac{}{\Phi \vdash \tau : \kappa \rhd (\!|\,|\!) \dashv \Phi} \qquad \texttt{RESVarHole}\quad \frac{\neg(t\ \texttt{valid})}{\Phi \vdash \tau : \kappa \rhd (\!|t|\!) \dashv \Phi}$$

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

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

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

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

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

(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* $\Delta; \Phi \vdash \tau : \kappa_2$ *for* $t : \texttt{Ty} \notin \Phi$ *then* $\Delta; \Phi \vdash \kappa_1 \lesssim \kappa_2$

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 a consistent supertype of the kappa synthesized by elaboration. The interesting rules in the kind assignments are: `KASubsumption` and `KASelfRecognition`. `KASubsumption` holds because this rule only makes the kind a consistent supertype of what it was. `KASelfRecognition` holds because elaboration always synthesizes the singleton-version of the kind for every type when possible. We need the side-condition to protect agaisnt `Ty` deriving from elaboration and $\texttt{S}(t)$ from `KASelfRecognition`.