

BinOp  $\oplus ::= \text{Product} \mid \text{Sum} \mid \text{Arrow}$   
 Kind  $\kappa ::= \text{Ty} \mid \text{KHole} \mid \text{S}(\tau)$   
 ConstantTypes  $c ::= \text{Int} \mid \text{Float} \mid \text{Bool}$   
 UserHTyp  $\hat{\tau} ::= c \mid \hat{\tau}_1 \oplus \hat{\tau}_2 \mid \text{list}(\hat{\tau}) \mid \langle \rangle^u \mid \langle \hat{\tau} \rangle^u$   
 InternalHTyp  $\tau ::= c \mid \tau_1 \oplus \tau_2 \mid \text{list}(\tau) \mid \langle \rangle^u \mid \langle \tau \rangle^u$   
 TypeVars  $t$   
 TypePattern  $\rho ::= t \mid \langle \rangle \mid \langle t \rangle$   
 UserExpression  $e ::= \text{type } \rho = \hat{\tau} \text{ in } e \mid \text{elided}$   
 InternalExpression  $\tau ::= \text{type } \rho = \tau : \kappa \text{ in } d \mid \text{elided}$

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

$\text{KHoleL}$   
 $\frac{}{\Delta; \Phi \vdash \text{KHole} \lesssim \kappa}$

$\text{KHoleR}$   
 $\frac{}{\Delta; \Phi \vdash \kappa \lesssim \text{KHole}}$

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

*KCSTVar*

$\frac{t :: k \in \Phi}{\Delta_j \Phi \vdash \text{S}(t) \lesssim k}$

$\text{KCSubsumption}$   
 $\frac{\Delta; \Phi \vdash \tau \Leftarrow \text{Ty}}{\Delta; \Phi \vdash \text{S}(\tau) \lesssim \text{Ty}}$   *$\tau$  not type variable?*

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

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

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

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

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

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

$$\begin{array}{c}
\text{KESym} \\
\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_2 \equiv \kappa_1} \\
\\
\text{KESingEquiv} \\
\frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash S(\tau_1) \equiv S(\tau_2)} \\
\\
\text{KESym} \quad \text{KETrans} \\
\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2 \quad \Delta; \Phi \vdash \kappa_2 \equiv \kappa_3}{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_3} \\
\\
\text{KESym} \\
\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_2 \equiv \kappa_1} \\
\\
\text{KESym} \\
\frac{\Delta; \Phi \vdash \kappa_1 \equiv \kappa_2}{\Delta; \Phi \vdash \kappa_2 \equiv \kappa_1}
\end{array}$$

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

$$\begin{array}{c}
\text{KSConst} \\
\frac{}{\Delta; \Phi \vdash c \Rightarrow S(c)} \\
\\
\text{KSVar} \\
\frac{t : \kappa \in \Phi}{\Delta; \Phi \vdash t \Rightarrow S(t)} \\
\\
\text{KSUVar} \\
\frac{t \notin \text{dom}(\Phi)}{\Delta; \Phi \vdash t \Rightarrow \text{KHole}} \\
\\
\text{KSBinOp} \\
\frac{\Delta; \Phi \vdash \tau_1 \Leftarrow S(\tau_1) \quad \Delta; \Phi \vdash \tau_2 \Leftarrow S(\tau_2)}{\Delta; \Phi \vdash \tau_1 \oplus \tau_2 \Rightarrow S(\tau_1 \oplus \tau_2)} \\
\\
\text{KSList} \\
\frac{\Delta; \Phi \vdash \tau \Leftarrow S(\tau)}{\Delta; \Phi \vdash \text{list}(\tau) \Rightarrow S(\text{list}(\tau))} \\
\\
\text{KSEHole} \\
\frac{u :: \kappa \in \Delta}{\Delta; \Phi \vdash \text{Hole}^u \Rightarrow \kappa} \\
\\
\text{KSNEHole} \\
\frac{u :: \kappa \in \Delta \quad \Delta; \Phi \vdash \tau \Rightarrow \kappa'}{\Delta; \Phi \vdash \text{NEHole}^u \Rightarrow \kappa}
\end{array}$$

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

$$\begin{array}{c}
\text{KAASubsume} \\
\frac{\Phi \vdash \tau \Rightarrow \kappa' \quad \Delta; \Phi \vdash \kappa' \lesssim \kappa}{\Delta; \Phi \vdash \tau \Leftarrow \kappa}
\end{array}$$

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

$$\begin{array}{c}
\text{KCESymm} \\
\frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash \tau_2 \equiv \tau_1} \\
\\
\text{KCETrans} \\
\frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 \quad \Delta; \Phi \vdash \tau_2 \equiv \tau_3}{\Delta; \Phi \vdash \tau_1 \equiv \tau_3} \\
\\
\text{KCESingEquiv} \\
\frac{\Delta; \Phi \vdash \tau_1 \Leftarrow \mathsf{S}(\tau_2)}{\Delta; \Phi \vdash \tau_1 \equiv \tau_2} \\
\\
\text{KCEConst} \\
\frac{}{\Delta; \Phi \vdash c \equiv c} \\
\\
\text{KCEVar} \\
\frac{t : \kappa \in \Phi}{\Delta; \Phi \vdash t \equiv t} \\
\\
\text{KCEBinOp} \\
\frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2 \quad \Delta; \Phi \vdash \tau_3 \equiv \tau_4}{\Delta; \Phi \vdash \tau_1 \oplus \tau_3 \equiv \tau_2 \oplus \tau_4} \\
\\
\text{KCEList} \\
\frac{\Delta; \Phi \vdash \tau_1 \equiv \tau_2}{\Delta; \Phi \vdash \text{list}(\tau_1) \equiv \text{list}(\tau_2)} \\
\\
\text{KCEEHole} \\
\frac{u :: \kappa \in \Delta}{\Delta; \Phi \vdash \langle \rangle^u \equiv \langle \rangle^u} \\
\\
\text{KCENEHole} \\
\frac{u :: \kappa \in \Delta \quad \Delta; \Phi \vdash \tau \Leftarrow \kappa'}{\Delta; \Phi \vdash \langle \tau \rangle^u \equiv \langle \tau \rangle^u}
\end{array}$$

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

TElabSConst

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

TElabSBinOp

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

TElabSList

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

TElabSVar

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

TElabSUNVar

$$\frac{t \notin \text{dom}(\Phi)}{\Phi \vdash t \Rightarrow \text{KHole} \rightsquigarrow (\textcolor{violet}{t})^u \dashv u :: \text{KHole}}$$

TElabSHole

$$\frac{}{\Phi \vdash (\textcolor{violet}{\text{Hole}})^u \Rightarrow \text{KHole} \rightsquigarrow (\textcolor{violet}{\text{Hole}})^u \dashv u :: \text{KHole}}$$

TElabSNEHole

$$\frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash (\hat{\tau})^u \Rightarrow \text{KHole} \rightsquigarrow (\textcolor{violet}{\tau})^u \dashv \Delta, u :: \text{KHole}}$$

$\boxed{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$   $\hat{\tau}$  analyzes against kind  $\kappa_1$  and elaborates to  $\tau$

TElabASubsume

$$\frac{\hat{\tau} \neq (\textcolor{violet}{\text{Hole}})^u \quad \hat{\tau} \neq (\hat{\tau}')^u \quad \Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta \quad \Delta; \Phi \vdash \kappa' \lesssim \kappa}{\Phi \vdash \hat{\tau} \Leftarrow \kappa \rightsquigarrow \tau \dashv \Delta}$$

TElabAEHole

$$\frac{}{\Phi \vdash (\textcolor{violet}{\text{Hole}})^u \Leftarrow \kappa \rightsquigarrow (\textcolor{violet}{\text{Hole}})^u \dashv u :: \kappa}$$

TElabANEHole

$$\frac{\Phi \vdash \hat{\tau} \Rightarrow \kappa' \rightsquigarrow \tau \dashv \Delta}{\Phi \vdash (\hat{\tau})^u \Leftarrow \kappa \rightsquigarrow (\textcolor{violet}{\tau})^u \dashv \Delta, u :: \kappa}$$

$\boxed{\Phi_1 \vdash \tau : \kappa \triangleright \rho \dashv \Phi_2}$   $\rho$  matches against  $\tau : \kappa$  extending  $\Phi$  if necessary

RESVar

$$\frac{t \text{ valid}}{\Phi \vdash \tau : \kappa \triangleright t \dashv \Phi, t :: \kappa}$$

RESEHole

$$\frac{}{\Phi \vdash \tau : \kappa \triangleright (\textcolor{blue}{\text{Hole}}) \dashv \Phi}$$

RESVarHole

$$\frac{\neg(t \text{ valid})}{\Phi \vdash \tau : \kappa \triangleright (\textcolor{blue}{t}) \dashv \Phi}$$

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

**ESDefine**

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

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

**DEDefine**

$$\frac{\Phi_1 \vdash \tau_1 : \kappa \triangleright \rho \dashv \Phi_2 \quad \Delta; \Gamma; \Phi_2 \vdash d : \tau_2}{\Delta; \Gamma; \Phi_1 \vdash \mathbf{type} \ \rho = \tau_1 : \kappa \ \mathbf{in} \ d : \tau_2}$$

### 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  $\Delta$  that is emitted from elaboration and then there's an  $\hat{\tau}$  that elaborates to any of the  $\tau$  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.