termvar, x	term variable
variant, V	variant
typvar, X	type variable
exc, *Exc*	exception
tyclass, C	typeclass name
tyclassvar, z	typeclass variable
n	
m	

program	::=			program
	|			nothing
	|	$top_1 \,..\, top_n \; program$		program
top	::=			toplevel construct
	|	*letBinding*		let binding
	|	**type alias** $X = T$		type alias
	|	**type** $X \; variantArgs = V_1 \; tyList_1 \,..\, \mid V_n \; tyList_n$		variants
	|	**exception** $Exc \; tyList$		exception
	|	**class** $C \; XKList =$ **let** $x_1 : T_1 \,..\,$ **let** $x_n : T_n$ **end**		typeclass definition
	|	**instance** $C \; TList = letBinding_1 \,..\, letBinding_n$ **end**		typeclass instance
	|	**instance** $[z] C \; TList = letBinding_1 \,..\, letBinding_n$ **end**		typeclass named instance
variantArgs	::=			
	|	$(X_1 : K_1) \,..\, (X_n : K_n)$		
tyList	::=			
	|	$T_1 \,..\, T_n$		
letBinding	::=			
	|	**let** $x \; params = t$		let binding
	|	**let rec** $x \; params = abs$		recursive let binding
t	::=			term
	|	x		variable
	|	V		type constructors
	|	$\lambda(x : T) \to t$		abstraction
	|	$\lambda(X : K) \to t$		type abstraction
	|	$\lambda?(z : C \; tyclassArgs) \to t$		typeclass abstraction
	|	$\lambda params \to t$	S	
	|	$t \; t'$		application
	|	$t[T]$		type application
	|	$t?[z]$		named typeclass applicatic
	|	$t?[C \; TList]$		typeclass application
	|	**let** $x \; params = t_1$ **in** t_2		let binding
	|	**let rec** $x \; params = abs$ **in** t		recursive let binding
	|	**match** t **with** $p_1 \to t_1 \,..\, \mid p_n \to t_n$ **end**		pattern matching
	|	$t : annot$		type annotation
	|	**fail** $[T] Exc \; t_1 \,..\, t_n$		fail
	|	**try** t **with** $pe_1 \to t_1 \,..\, \mid pe_n \to t_n$ **end**		try
	|	$t; t'$	S	== let _ : Unit = t in t'
	|	(t)	S	
	|	**failure** *exnval*		
	|	**TConstr** $V \; v_1 \,..\, v_n$		
	|	$t.x$		
v	::=			value

	$\mid$	$\mathbf{TConstr}\ V\ v_1\ ..\ v_n$	type constructors
	$\mid$	$\lambda(x : T) \to t$	abstraction
	$\mid$	$\lambda?(z : C\ tyclassArgs) \to t$	typeclass abstraction
	$\mid$	$\lambda valueParams \to v$	S
	$\mid$	$\mathbf{let\ rec}\ x\ params = valAbs_1\ \mathbf{in}\ valAbs_2$	recursive let binding
valAbs	::=		
	$\mid$	$\lambda(x : T) \to t$	abstraction
	$\mid$	$\lambda?(z : C\ tyclassArgs) \to t$	typeclass abstraction
abs	::=		lambda abstractions
	$\mid$	$\lambda(x : T) \to t$	abstraction
	$\mid$	$\lambda(X : K) \to abs$	type abstraction
	$\mid$	$\lambda?(z : C\ tyclassArgs) \to t$	typeclass abstraction
	$\mid$	$\mathbf{let}\ x\ params = t\ \mathbf{in}\ abs$	let binding
	$\mid$	$\mathbf{let\ rec}\ x\ params = abs_1\ \mathbf{in}\ abs_2$	recursive let binding
	$\mid$	$abs : annot$	type annotation
eff	::=		effect
	$\mid$	$effelm_1, .., effelm_n$	
	$\mid$	(eff)	S
effelm	::=		effects elements
	$\mid$	X	effect
	$\mid$	**IO**	IO effect
	$\mid$	$\mathbf{Exn}\,[exn]$	exception
exn	::=		exceptions
	$\mid$	$Exc_1 \mid .. \mid Exc_n$	
K	::=		kinds
	$\mid$	$*$	star
	$\mid$	φ	the effect kind
	$\mid$	$K \to K'$	kind arrow
T	::=		type
	$\mid$	X	variable
	$\mid$	$[eff]$	effects
	$\mid$	**Unit**	Unit type (contained in the module opened by
	$\mid$	$T \to T'$	S == T -[]-¿ T'
	$\mid$	$T - [eff] -> T'$	function
	$\mid$	$\lambda(X : K), T$	operator abstraction
	$\mid$	$\forall\,(X : K), T$	forall
	$\mid$	$\forall\, tyParams, T$	S
	$\mid$	$\{C\ tyclassArgs\} => T$	S == C tyclassArgs =[]=¿
	$\mid$	$\{C\ tyclassArgs\} = [eff] => T$	typeclass
	$\mid$	$T\ T'$	operator application

	|	(T)	S
	|	$T_1 \to .. \to T_n \to T$	S
	|	$T\ T_1 .. T_n$	S
p	::=		pattern
	|	$V\ p_1 .. p_n$	variant
	|	x	wildcard variable
pe	::=		try pattern
	|	$Exc\ x_1 .. x_n$	Exception pattern
$lambda$	::=		lambda parameters
	|	$(x : T)$	value
	|	$tyLambda$	type
$valueLambda$	::=		
	|	$(x : T)$	value
$tyLambda$	::=		
	|	$(X : K)$	type
	|	X	S
$valueParams$	::=		
	|	$valueLambda_1 .. valueLambda_n$	S
$tyParams$	::=		
	|	$tyLambda_1 .. tyLambda_n$	S
$exnval$	::=		runtime value of exceptions
	|	$Exc\ v_1 .. v_n$	
$tyclassArg$	::=		typeclass argument
	|	X	type variable
	|	$[T]$	type
$resolvedTyclassArg$	::=		Resolved typeclass argument
	|	$[T]$	type
$tyclassArgs$	::=		typeclass arguments
	|	$tyclassArg_1 .. tyclassArg_n$	arguments
$resolvedTyclassArgs$	::=		Resolved typeclass arguments
	|	$resolvedTyclassArg_1 .. resolvedTyclassArg_n$	arguments
$annot$	::=		
	|	T	S
	|	$[[eff]]\,T$	S

params	::=			
	|	$lambda_1 \,..\, lambda_n$	S	
	|	$lambda_1 \,..\, lambda_n : annot$	S	
Γ	::=			type environment
	|	$\emptyset$		empty
	|	$\Gamma, x_1 : T_1, \,.., x_n : T_n$		vars
	|	$\Gamma, V : T$	S	type constructors (contained in the above
	|	$\Gamma, X_1 : K_1, \,.., X_n : K_n$		tvars
	|	$\Gamma, X : \{ V_1\ tyList_1 \,..\, V_n\ tyList_n \}$		variants
	|	$\Gamma, Exc\ tyList$		exceptions
	|	$\Gamma, C\ XKList\ tyclassSigs$		typeclass
	|	$\Gamma, C\ TList \{ letBinding_1 \,..\, letBinding_n \}$		instance
	|	$\Gamma, z : C\ tyclassArgs$		named instance
Δ	::=			runtime environment
	|	$\emptyset$		empty
	|	$\Delta, \{ x_1 \leftarrow v_1 \,..\, x_n \leftarrow v_n \}$		vars
	|	$\Delta, \{ V_1 \leftarrow v_1 \,..\, V_n \leftarrow v_n \}$		variant vars
tyclassSigs	::=			
	|	$\{ x_1 : T_1 \,..\, x_n : T_n \}$		
patterns	::=			
	|	$p_1 \,..\, p_n$		
VArgs	::=			
	|	$T_1 \,..\, T_n$		
Variant	::=			
	|	$V_1\ VArgs_1 \,..\, V_n\ VArgs_n$		
RetVar	::=			
	|	$X\ X_1 \,..\, X_n$		
XKList	::=			
	|	$X_1 : K_1 \,..\, X_n : K_n$		
TList	::=			
	|	$T_1 \,..\, T_n$		

$\boxed{\Gamma \vdash program \rhd \Gamma'}$ Program typing

$$\frac{}{\Gamma \vdash \rhd \Gamma} \textsc{Prog\_Empty}$$

$$\frac{\Gamma \vdash top \rhd \Gamma' \quad \Gamma' \vdash program \rhd \Gamma''}{\Gamma \vdash top\ program \rhd \Gamma''} \textsc{Prog\_Program}$$

$\boxed{\Gamma \vdash top \rhd \Gamma'}$ Toplevel typing

$$\frac{\Gamma \vdash \mathit{letBinding} \rhd x : T}{\Gamma \vdash \mathit{letBinding} \rhd \Gamma, x : T} \quad \textsc{Top\_Let}$$

$$\frac{\Gamma \vdash T : K}{\Gamma \vdash \textbf{type alias}\ X = T \rhd \Gamma, X : K} \quad \textsc{Top\_TypeAlias}$$

$$\frac{\begin{array}{l}\mathit{RetVar} = X\ X_1\ ..\ X_n \\ \Gamma' = \Gamma, X : K, X_1 : K_1, .., X_n : K_n \\ \mathit{RetVar}\ \&\ \Gamma' \vdash V_1\ \mathit{tyList}_1 \rhd \Gamma_1 \quad .. \quad \mathit{RetVar}\ \&\ \Gamma' \vdash V_n\ \mathit{tyList}_n \rhd \Gamma_n\end{array}}{\Gamma \vdash \textbf{type}\ X\ (X_1 : K_1)(X_n : K_n) = V_1\ \mathit{tyList}_1 | .. | V_n\ \mathit{tyList}_n \rhd \Gamma \cup \Gamma_1 \cup .. \cup \Gamma_n, X : \{V_1\ \mathit{tyList}_1\ ..\ V_n\ \mathit{tyList}_n\}} \quad \textsc{T}$$

$$\frac{\Gamma \vdash T_1 : K_1 \quad .. \quad \Gamma \vdash T_n : K_n}{\Gamma \vdash \textbf{exception}\ \mathit{Exc}\ T_1\ ..\ T_n \rhd \Gamma, \mathit{Exc}\ T_1\ ..\ T_n} \quad \textsc{Top\_Exception}$$

$$\frac{\begin{array}{l}\Gamma \vdash T_1 : K_1 \quad .. \quad \Gamma \vdash T_n : K_n \\ T'_1 \equiv \{C\ X_1\ ..\ X_n\} => T_1 \quad .. \quad T'_n \equiv \{C\ X_1\ ..\ X_n\} => T_n\end{array}}{\Gamma \vdash \textbf{class}\ C\ X_1 : K_1\ ..\ X_n : K_n = \textbf{let}\ x_1 : T_1\ ..\ \textbf{let}\ x_n : T_n\ \textbf{end} \rhd \Gamma, C\ X_1 : K_1\ ..\ X_n : K_n\ \{x_1 : T_1\ ..\ x_n : T_n\} \cup \emptyset, x_1}$$

$$\frac{\begin{array}{l}C\ \mathit{XKList}\ \mathit{tyclassSigs}\ \in \Gamma \\ \Gamma \vdash \mathit{TList}\ \mathit{tyclassSigs}\ \textbf{matches}\ \mathit{XKList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\}\end{array}}{\Gamma \vdash \textbf{instance}\ C\ \mathit{TList} = \mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\ \textbf{end} \rhd \Gamma, C\ \mathit{TList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\}} \quad \textsc{Top\_Instanc}$$

$$\frac{\begin{array}{l}C\ \mathit{XKList}\ \mathit{tyclassSigs}\ \in \Gamma \\ \Gamma \vdash \mathit{TList}\ \mathit{tyclassSigs}\ \textbf{matches}\ \mathit{XKList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\} \\ \mathit{TList} \equiv \mathit{resolvedTyclassArgs}\end{array}}{\Gamma \vdash \textbf{instance}\ [z]C\ \mathit{TList} = \mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\ \textbf{end} \rhd \Gamma, C\ \mathit{TList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\} \cup \emptyset, z : C\ \mathit{reso}$$

$\boxed{\Gamma \vdash \mathit{letBinding} \rhd x : T}$ Toplevel let binding typing

$$\frac{\Gamma \vdash t : [[\,]]\,T}{\Gamma \vdash \textbf{let}\ x\ = t \rhd x : T} \quad \textsc{TyLetBinding\_Let}$$

$$\frac{\Gamma, x : T \vdash \mathit{abs} : [[\,]]\,T}{\Gamma \vdash \textbf{let rec}\ x\ : T = \mathit{abs} \rhd x : T} \quad \textsc{TyLetBinding\_LetRec}$$

$\boxed{\Gamma \vdash \mathit{TList}\ \mathit{tyclassSigs}\ \textbf{matches}\ \mathit{XKList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\}}$ Typeclass instances matching

$$\frac{\begin{array}{l}\Gamma \vdash \mathit{letBinding}_1 \rhd x_1 : T'_1 \quad .. \quad \Gamma \vdash \mathit{letBinding}_n \rhd x_n : T'_n \\ \Gamma \vdash [\mathit{XKList} \mapsto \mathit{TList}]\,T_1 \rhd T'_1 \quad .. \quad \Gamma \vdash [\mathit{XKList} \mapsto \mathit{TList}]\,T_n \rhd T'_n\end{array}}{\Gamma \vdash \mathit{TList}\ \{x_1 : T_1\ ..\ x_n : T_n\}\ \textbf{matches}\ \mathit{XKList}\{\mathit{letBinding}_1\ ..\ \mathit{letBinding}_n\}} \quad \textsc{TyClassInstanceMatches\_LetB}$$

$\boxed{\Gamma \vdash [\mathit{XKList} \mapsto \mathit{TList}]\,T \rhd T'}$ Type substitution in instances

$$\frac{\begin{array}{c}\Gamma \vdash T_1 : K_1 \quad .. \quad \Gamma \vdash T_n : K_n \\ [X_1 \mapsto T_1\ ..\ X_n \mapsto T_n]\,T \equiv T'\end{array}}{\Gamma \vdash [X_1 : K_1\ ..\ X_n : K_n \mapsto T_1\ ..\ T_n]\,T \rhd T'} \quad \textsc{TySubstituteInstance\_Sub}$$

$\boxed{\mathit{RetVar}\ \&\ \Gamma \vdash V\ \mathit{tyList} \rhd \Gamma'}$ Type declaration

$$\frac{\Gamma \vdash T_1 : K_1 \quad .. \quad \Gamma \vdash T_n : K_n}{X\ X_1\ ..\ X_n\ \&\ \Gamma \vdash V\ T_1\ ..\ T_n \rhd \emptyset, V : T_1 \to .. \to T_n \to X\ X_1\ ..\ X_n} \quad \textsc{TyDecl\_Decl}$$

$\boxed{\Gamma \vdash t : [[\mathit{eff}]]\,T}$ Typing

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : [[\,]]\,T} \quad \textsc{T\_Var}$$

$$\frac{V : T \in \Gamma}{\Gamma \vdash V : [[]]\,T} \quad \text{T\_Variant}$$

$$\frac{\Gamma \vdash t : [[\mathit{eff}_1]]\{C\ \mathit{resolvedTyclassArgs}\} = [\mathit{eff}_2] => T}{\Gamma \vdash t : [[\mathit{eff}_1 \cup \mathit{eff}_2]]\,T} \quad \text{T\_TyClassElim}$$

$$\frac{\begin{array}{c}\Gamma, x_1 : T_1 \vdash t : [[\mathit{eff}]]\,T \\ \Gamma \vdash T_1 : *\end{array}}{\Gamma \vdash \lambda(x_1 : T_1) \to t : [[]]\,T_1 - [\mathit{eff}] -> T} \quad \text{T\_Abs}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : [[\mathit{eff}_1]]\{C_1\ \mathit{tyclassArgs}_1\} = [\mathit{eff}_1'] => .. \{C_n\ \mathit{tyclassArgs}_n\} = [\mathit{eff}_n'] => T_1 - [\mathit{eff}_2] -> T_2 \\ \textbf{unify type variables from } \mathit{tyclassArgs}_1 .. \mathit{tyclassArgs}_n \textbf{ in } T1, \mathit{eff}_2 \textbf{ and } T_2 \rhd eff2'', T_2' \\ \Gamma \vdash t' : [[\mathit{eff}_3]]\,T_1\end{array}}{\Gamma \vdash t\ t' : [[\mathit{eff}_1 \cup \mathit{eff}_2'' \cup \mathit{eff}_3]]\{C\ \mathit{tyclassArgs}_1'\} = [\mathit{eff}_1'] => .. \{C\ \mathit{tyclassArgs}_n'\} = [\mathit{eff}_n'] => T_2'} \quad \text{T\_App}$$

$$\frac{\begin{array}{l}\Gamma, X : K \vdash t : [[\mathit{eff}]]\,T \\ \textbf{IO} \notin \mathit{eff}\end{array}}{\Gamma \vdash \lambda(X : K) \to t : [[\mathit{eff}]] \forall\,(X : K), T} \quad \text{T\_TAbs}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : [[\mathit{eff}]] \forall\,(X : K), T_2 \\ \Gamma \vdash T_1 : K\end{array}}{\Gamma \vdash t[T_1] : [[\mathit{eff}]][X \mapsto T_1]\,T_2} \quad \text{T\_TApp}$$

$$\frac{\begin{array}{l}\Gamma, z : C\ \mathit{tyclassArgs} \vdash t : [[\mathit{eff}]]\,T \\ C\ X_1 : K_1 .. X_n : K_n\ \mathit{tyclassSigs} \in \Gamma \\ \Gamma \vdash \mathit{tyclassArg}_1 : K_1 \quad .. \quad \Gamma \vdash \mathit{tyclassArg}_n : K_n\end{array}}{\Gamma \vdash \lambda?(z : C\ \mathit{tyclassArg}_1 .. \mathit{tyclassArg}_n) \to t : [[]]\{C\ \mathit{tyclassArg}_1 .. \mathit{tyclassArg}_n\} = [\mathit{eff}] => T} \quad \text{T\_CAbs}$$

$$\frac{\begin{array}{l}z : C\ \mathit{tyclassArgs} \in \Gamma \\ \Gamma \vdash t : [[\mathit{eff}_1]]\{C\ \mathit{tyclassArgs}\} = [\mathit{eff}_2] => T\end{array}}{\Gamma \vdash t?[z] : [[\mathit{eff}_1 \cup \mathit{eff}_2]]\,T} \quad \text{T\_CApp1}$$

$$\frac{\begin{array}{l}C\ \mathit{TList}\{\mathit{letBinding}_1 .. \mathit{letBinding}_n\} \in \Gamma \\ \Gamma \vdash t : [[\mathit{eff}_1]]\{C\ \mathit{tyclassArgs}\} = [\mathit{eff}_2] => T \\ \mathit{TList} \textbf{ matches } \mathit{tyclassArgs}\end{array}}{\Gamma \vdash t?[C\ \mathit{TList}] : [[\mathit{eff}_1 \cup \mathit{eff}_2]]\,T} \quad \text{T\_CApp2}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : [[\mathit{eff}]]\,X \\ X \equiv X' \\ \Gamma \vdash X' : *\end{array}}{\Gamma \vdash t : [[\mathit{eff}]]\,X'} \quad \text{T\_Eq}$$

$$\frac{\Gamma, x : T_1 \vdash t_2 : [[\mathit{eff}_2]]\,T_2}{\Gamma \vdash \textbf{let}\ x\ = (t_1 : [[\mathit{eff}_1]]\,T_1) \textbf{ in } t_2 : [[\mathit{eff}_1 \cup \mathit{eff}_2]]\,T_2} \quad \text{T\_Let}$$

$$\frac{\begin{array}{c}\Gamma, x : T_1 \vdash \mathit{abs} : [[\mathit{eff}_1]]\,T_1 \\ \Gamma, x : T_1 \vdash t_2 : [[\mathit{eff}_2]]\,T_2\end{array}}{\Gamma \vdash \textbf{let rec}\ x\ = \mathit{abs} : [[\mathit{eff}_1]]\,T_1 \textbf{ in } t_2 : [[\mathit{eff}_1 \cup \mathit{eff}_2]]\,T_2} \quad \text{T\_LetRec}$$

$$\frac{\begin{array}{l}\Gamma \cup \Gamma_1 \vdash t_1 : [[\mathit{eff}_1]]\,T_2 \quad .. \quad \Gamma \cup \Gamma_n \vdash t_n : [[\mathit{eff}_n]]\,T_2 \\ T_1 : \{\mathit{Variant}\} \in \Gamma \\ \mathit{Variant}\ \&\ \Gamma \vdash p_1 .. p_n : T_1 \rhd \Gamma_1 .. \Gamma_n \\ \Gamma \vdash t : [[\mathit{eff}]]\,T_1\end{array}}{\Gamma \vdash \textbf{match}\ t\ \textbf{with}\ p_1 \to t_1 | .. | p_n \to t_n\ \textbf{end} : [[\mathit{eff}\ \cup\ \mathit{eff}_1 \cup .. \cup\ \mathit{eff}_n]]\,T_2} \quad \text{T\_Match}$$

$$\frac{\begin{array}{l}\Gamma \vdash t : [[\mathit{eff}]]\,T \\ \Gamma \vdash T : * \\ \Gamma \vdash \mathit{eff} : \varphi\end{array}}{\Gamma \vdash (t : [[\mathit{eff}]]\,T) : [[\mathit{eff}]]\,T}\ \text{T\_Annot}$$

$$\frac{\begin{array}{l}\Gamma \vdash t_1 : [[\mathit{eff}_1]]\,T_1 \quad .. \quad \Gamma \vdash t_n : [[\mathit{eff}_n]]\,T_n \\ \Gamma \vdash T : * \\ \mathit{Exc}\ T_1\ ..\ T_n\ \in \Gamma\end{array}}{\Gamma \vdash \mathbf{fail}\,[T]\mathit{Exc}\ t_1\ ..\ t_n : [[\mathbf{Exn}\,[\mathit{Exc}]\ \cup\ \mathit{eff}_1\ \cup\ ..\ \cup\ \mathit{eff}_n]]\,T}\ \text{T\_Fail}$$

$$\frac{\begin{array}{l}\Gamma_1 \vdash t_1 : [[\mathit{eff}_1]]\,T \quad .. \quad \Gamma_n \vdash t_n : [[\mathit{eff}_n]]\,T \\ \Gamma \vdash \mathit{pe}_1 \rhd \mathit{Exc}_1\ \&\ \Gamma_1 \quad .. \quad \Gamma \vdash \mathit{pe}_n \rhd \mathit{Exc}_n\ \&\ \Gamma_n \\ \Gamma \vdash t : [[\mathit{eff}]]\,T\end{array}}{\Gamma \vdash \mathbf{try}\ t\ \mathbf{with}\ \mathit{pe}_1 \to t_1 |\ ..\ |\mathit{pe}_n \to t_n\ \mathbf{end} : [[(\mathit{eff} \setminus [\mathit{Exc}_1|\ ..\ |\mathit{Exc}_n]) \cup \mathit{eff}_1 \cup\ ..\ \cup \mathit{eff}_n]]\,T}\ \text{T\_Try}$$

$\boxed{\mathit{TList}\ \mathbf{matches}\ \mathit{tyclassArgs}}$ Type list matches typeclass arguments

$$\frac{}{T\ \mathbf{matches}\ X}\ \text{MatchAll}$$

$$\frac{}{T\ \mathbf{matches}\,[T]}\ \text{MatchType}$$

$$\frac{T_1\ \mathbf{matches}\ \mathit{tyclassArg}_1 \quad .. \quad T_n\ \mathbf{matches}\ \mathit{tyclassArg}_n}{T_1\ ..\ T_n\ \mathbf{matches}\ \mathit{tyclassArg}_1\ ..\ \mathit{tyclassArg}_n}\ \text{RecRule}$$

$\boxed{\Gamma \vdash T : K}$ Kinding

$$\frac{X : K\ \in \Gamma}{\Gamma \vdash X : K}\ \text{K\_TVar}$$

$$\frac{\Gamma \vdash \mathit{eff} : \varphi}{\Gamma \vdash [\mathit{eff}] : \varphi}\ \text{K\_Eff}$$

$$\frac{\Gamma, X : K_1 \vdash T : K_2}{\Gamma \vdash \lambda(X : K_1), T : K_1 \to K_2}\ \text{K\_Abs}$$

$$\frac{\begin{array}{l}\Gamma \vdash T_1 : K_{11} \to K_{12} \\ \Gamma \vdash T_2 : K_{11}\end{array}}{\Gamma \vdash T_1\ T_2 : K_{12}}\ \text{K\_App}$$

$$\frac{\begin{array}{l}\Gamma \vdash T_1 : * \\ \Gamma \vdash \mathit{eff} : \varphi \\ \Gamma \vdash T_2 : *\end{array}}{\Gamma \vdash T_1 - [\mathit{eff}] -> T_2 : *}\ \text{K\_Arrow}$$

$$\frac{\Gamma, X : K_1 \vdash T_2 : *}{\Gamma \vdash \forall\,(X : K_1), T_2 : *}\ \text{K\_All}$$

$$\frac{\begin{array}{l}C\ X_1 : K_1\ ..\ X_n : K_n\ \mathit{tyclassSigs}\ \in \Gamma \\ \Gamma \vdash \mathit{tyclassArg}_1 : K_1 \quad .. \quad \Gamma \vdash \mathit{tyclassArg}_n : K_n \\ \Gamma \vdash \mathit{eff} : \varphi \\ \Gamma \vdash T : *\end{array}}{\Gamma \vdash \{C\ \mathit{tyclassArg}_1\ ..\ \mathit{tyclassArg}_n\} = [\mathit{eff}] => T : *}\ \text{K\_TyClass}$$

$\boxed{\Gamma \vdash \mathit{tyclassArg} : K}$ Typeclass argument kinding

$$\frac{}{\Gamma \vdash X : K}\ \text{TyClassArgKind\_Variable}$$

$$\frac{\Gamma \vdash T : K}{\Gamma \vdash [T] : K} \quad \textsc{TyClassArgKind\_Type}$$

$\boxed{\Gamma \vdash \mathit{eff} : \varphi}$ Effects typing

$$\frac{\Gamma \vdash \mathit{effelm}_1 : \varphi \quad .. \quad \Gamma \vdash \mathit{effelm}_n : \varphi}{\Gamma \vdash \mathit{effelm}_1, .., \mathit{effelm}_n : \varphi} \quad \textsc{Eff\_Eff}$$

$\boxed{\Gamma \vdash \mathit{effelm} : \varphi}$ Effects elements typing

$$\frac{X : \varphi \in \Gamma}{\Gamma \vdash X : \varphi} \quad \textsc{EffElm\_Eff}$$

$$\frac{}{\Gamma \vdash \mathbf{IO} : \varphi} \quad \textsc{EffElm\_IO}$$

$$\frac{\mathit{Exc}_1\ \mathit{tyList}_1 \in \Gamma \quad .. \quad \mathit{Exc}_n\ \mathit{tyList}_n \in \Gamma}{\Gamma \vdash \mathbf{Exn}\,[\mathit{Exc}_1|\,..\,|\mathit{Exc}_n] : \varphi} \quad \textsc{EffElm\_Exn}$$

$\boxed{\mathit{Variant}\ \&\ \Gamma \vdash \mathit{patterns} : T \rhd \Gamma_1\,..\,\Gamma_n}$ Patterns matching typing

$$\frac{\mathit{Variant}\ \&\ \Gamma \vdash p_1 : T \rhd \Gamma_1 \quad .. \quad \mathit{Variant}\ \&\ \Gamma \vdash p_n : T \rhd \Gamma_n}{\mathit{Variant}\ \&\ \Gamma \vdash p_1\,..\,p_n : T \rhd \Gamma_1\,..\,\Gamma_n} \quad \textsc{PsTy\_Patterns}$$

$\boxed{\mathit{Variant}\ \&\ \Gamma \vdash p : T \rhd \Gamma'}$ Pattern matching typing

$$\frac{\begin{array}{l} V \in \mathit{Variant} \rhd T_1\,..\,T_n \\ T_1 : \{\mathit{Variant}_1\} \in \Gamma \quad .. \quad T_n : \{\mathit{Variant}_n\} \in \Gamma \\ \mathit{Variant}_1\ \&\ \Gamma \vdash p_1 : T_1 \rhd \Gamma_1 \quad .. \quad \mathit{Variant}_n\ \&\ \Gamma \vdash p_n : T_n \rhd \Gamma_n \end{array}}{\mathit{Variant}\ \&\ \Gamma \vdash V\ p_1\,..\,p_n : T \rhd \Gamma_1 \cup .. \cup \Gamma_n} \quad \textsc{PTy\_Variant}$$

$$\frac{}{\mathit{Variant}\ \&\ \Gamma \vdash x : T \rhd \emptyset, x : T} \quad \textsc{PTy\_Wildcard}$$

$\boxed{\Gamma \vdash \mathit{pe} \rhd \mathit{Exc}\ \&\ \Gamma'}$ Exception pattern matching typing

$$\frac{\mathit{Exc}\ T_1\,..\,T_n \in \Gamma}{\Gamma \vdash \mathit{Exc}\ x_1\,..\,x_n \rhd \mathit{Exc}\ \&\ \Gamma, x_1 : T_1, .., x_n : T_n} \quad \textsc{PETy\_Exc}$$

$\boxed{T \equiv T'}$ Type equivalence

$$\frac{}{T \equiv T} \quad \textsc{Q\_Refl}$$

$$\frac{T \equiv T'}{T' \equiv T} \quad \textsc{Q\_Symm}$$

$$\frac{\begin{array}{c} T_1 \equiv T_2 \\ T_2 \equiv T_3 \end{array}}{T_1 \equiv T_3} \quad \textsc{Q\_Trans}$$

$$\frac{\begin{array}{c} T_{11} \equiv T_{21} \\ \mathit{eff}_1 \equiv \mathit{eff}_2 \\ T_{12} \equiv T_{22} \end{array}}{T_{11} - [\mathit{eff}_1] -> T_{12} \equiv T_{21} - [\mathit{eff}_2] -> T_{22}} \quad \textsc{Q\_Arrow}$$

$$\frac{T_1 \equiv T_2}{\forall\,(X : K), T_1 \equiv \forall\,(X : K), T_2} \quad \textsc{Q\_All}$$

$$\frac{\begin{array}{c}\mathit{eff}_1 \equiv \mathit{eff}_2 \\ T_1 \equiv T_2\end{array}}{\{C\ \mathit{tyclassArgs}\} = [\mathit{eff}_1] => T_1 \equiv \{C\ \mathit{tyclassArgs}\} = [\mathit{eff}_2] => T_2}\ \textsc{Q\_TyClass}$$

$$\frac{T_1 \equiv T_2}{\lambda(X : K), T_1 \equiv \lambda(X : K), T_2}\ \textsc{Q\_Abs}$$

$$\frac{\begin{array}{c}T_{11} \equiv T_{21} \\ T_{12} \equiv T_{22}\end{array}}{T_{11}\ T_{12} \equiv T_{21}\ T_{22}}\ \textsc{Q\_App}$$

$$\frac{}{(\lambda(X : K), T_{11})\ T_{12} \equiv [X \mapsto T_{12}]T_{11}}\ \textsc{Q\_AppAbs}$$

$\boxed{\mathit{eff} \equiv \mathit{eff}'}$ Effects equivalence

$$\frac{}{\mathit{eff} \equiv \mathit{eff}}\ \textsc{EffEq\_Refl}$$

$$\frac{\mathbf{set}\,(\mathit{effelm}_1, .., \mathit{effelm}_n) = \mathbf{set}\,(\mathit{effelm}'_1, .., \mathit{effelm}'_n)}{\mathit{effelm}_1, .., \mathit{effelm}_n \equiv \mathit{effelm}'_1, .., \mathit{effelm}'_n}\ \textsc{EffEq\_Eq}$$

$\boxed{\mathit{effelm} \equiv \mathit{effelm}'}$ Effect element equivalence

$$\frac{}{\mathit{effelm} \equiv \mathit{effelm}}\ \textsc{EffElmEq\_Refl}$$

$$\frac{\mathit{effelm} \equiv \mathit{effelm}'}{\mathit{effelm}' \equiv \mathit{effelm}}\ \textsc{EffElmEq\_Symm}$$

$$\frac{\begin{array}{c}\mathit{effelm}_1 \equiv \mathit{effelm}_2 \\ \mathit{effelm}_2 \equiv \mathit{effelm}_3\end{array}}{\mathit{effelm}_1 \equiv \mathit{effelm}_3}\ \textsc{EffElmEq\_Trans}$$

$$\frac{\mathbf{set}\,(\mathit{Exc}_1|..|\mathit{Exc}_n) = \mathbf{set}\,(\mathit{Exc}'_1|..|\mathit{Exc}'_n)}{\mathbf{Exn}\,[\mathit{Exc}_1|..|\mathit{Exc}_n] \equiv \mathbf{Exn}\,[\mathit{Exc}'_1|..|\mathit{Exc}'_n]}\ \textsc{EffElmEq\_ExnEq}$$

$\boxed{\Delta \vdash \mathit{program} \longrightarrow \Delta' \vdash \mathit{program}'}$ Toplevel evaluation

$$\frac{\Delta \vdash t \longrightarrow \Delta \vdash t'}{\Delta \vdash \mathbf{let}\,x\ = t\ \mathit{program} \longrightarrow \Delta \vdash \mathbf{let}\,x\ = t'\ \mathit{program}}\ \textsc{TopE\_Let1}$$

$$\frac{}{\Delta \vdash \mathbf{let}\,x\ = v\ \mathit{program} \longrightarrow \Delta, \{x \leftarrow v\} \vdash \mathit{program}}\ \textsc{TopE\_Let2}$$

$$\frac{\Delta \vdash \mathit{abs} \longrightarrow \Delta \vdash \mathit{abs}'}{\Delta \vdash \mathbf{let\,rec}\,x\ = \mathit{abs}\ \mathit{program} \longrightarrow \Delta \vdash \mathbf{let\,rec}\,x\ = \mathit{abs}'\ \mathit{program}}\ \textsc{TopE\_LetRec1}$$

$$\frac{}{\Delta \vdash \mathbf{let\,rec}\,x\ = \mathit{valAbs}\ \mathit{program} \longrightarrow \Delta, \{x \leftarrow \mathbf{let\,rec}\,x\ = \mathit{valAbs}\,\mathbf{in}\,\mathit{valAbs}\} \vdash \mathit{program}}\ \textsc{TopE\_LetRec2}$$

$$\frac{}{\Delta \vdash \mathbf{type\,alias}\,X = T\ \mathit{program} \longrightarrow \Delta \vdash \mathit{program}}\ \textsc{TopE\_TypeAlias}$$

$$\frac{V_1\ \mathit{tyList}_1 \rhd v_1 \quad .. \quad V_n\ \mathit{tyList}_n \rhd v_n}{\Delta \vdash \mathbf{type}\,X\ \mathit{variantArgs} = V_1\ \mathit{tyList}_1|..|V_n\ \mathit{tyList}_n\ \mathit{program} \longrightarrow \Delta, \{V_1 \leftarrow v_1 .. V_n \leftarrow v_n\} \vdash \mathit{program}}\ \textsc{TopE\_T}$$

$$\frac{}{\Delta \vdash \mathbf{exception}\,\mathit{Exc}\ \mathit{tyList}\ \mathit{program} \longrightarrow \Delta \vdash \mathit{program}}\ \textsc{TopE\_Exception}$$

$\boxed{V\ \mathit{tyList} \rhd v}$ Variants creation

$$\frac{(\{x_1 \leftarrow v_1\} \in \Delta)\,\textbf{after applications} \quad .. \quad (\{x_n \leftarrow v_n\} \in \Delta)\,\textbf{after applications}}{V\ T_1\ ..\ T_n \rhd \lambda(x_1 : T_1)\ ..\ (x_n : T_n) \to \textbf{TConstr}\ V\ v_1\ ..\ v_n}\ \textsc{VarCreation\_Create}$$

$\boxed{\Delta \vdash t \longrightarrow \Delta' \vdash t'}$ Evaluation

$$\frac{\{x \leftarrow v\} \in \Delta}{\Delta \vdash x \longrightarrow \Delta \vdash v}\ \textsc{E\_Var}$$

$$\frac{\{V \leftarrow v\} \in \Delta}{\Delta \vdash V \longrightarrow \Delta \vdash v}\ \textsc{E\_Variant}$$

$$\frac{}{\Delta \vdash (\textbf{failure}\ exnval)\ v \longrightarrow \Delta \vdash \textbf{failure}\ exnval}\ \textsc{E\_App1Failure}$$

$$\frac{}{\Delta \vdash t\ (\textbf{failure}\ exnval) \longrightarrow \Delta \vdash \textbf{failure}\ exnval}\ \textsc{E\_App2Failure}$$

$$\frac{\Delta \vdash t_1 \longrightarrow \Delta \vdash t_1'}{\Delta \vdash t\ t_1 \longrightarrow \Delta \vdash t\ t_1'}\ \textsc{E\_App1}$$

$$\frac{\Delta \vdash t_1 \longrightarrow \Delta \vdash t_1'}{\Delta \vdash t_1\ v \longrightarrow \Delta \vdash t_1'\ v}\ \textsc{E\_App2}$$

$$\frac{}{\Delta \vdash (\lambda(x : T) \to t_{12})\ v_2 \longrightarrow \Delta, \{x \leftarrow v_2\} \vdash t_{12}}\ \textsc{E\_AppAbs}$$

$$\frac{}{\Delta \vdash \lambda(X : K) \to t \longrightarrow \Delta \vdash t}\ \textsc{E\_TAbs}$$

$$\frac{}{\Delta \vdash t[T] \longrightarrow \Delta \vdash t}\ \textsc{E\_TApp}$$

$$\frac{}{\Delta \vdash \textbf{let}\ x\ = \textbf{failure}\ exnval\ \textbf{in}\ t_2 \longrightarrow \Delta \vdash \textbf{failure}\ exnval}\ \textsc{E\_LetFailure}$$

$$\frac{\Delta \vdash t_1 \longrightarrow \Delta \vdash t_1'}{\Delta \vdash \textbf{let}\ x\ = t_1\ \textbf{in}\ t_2 \longrightarrow \Delta \vdash \textbf{let}\ x\ = t_1'\ \textbf{in}\ t_2}\ \textsc{E\_Let1}$$

$$\frac{}{\Delta \vdash \textbf{let}\ x\ = v\ \textbf{in}\ t \longrightarrow \Delta, \{x \leftarrow v\} \vdash t}\ \textsc{E\_Let2}$$

$$\frac{\Delta \vdash abs \longrightarrow \Delta \vdash abs'}{\Delta \vdash \textbf{let rec}\ x\ = abs\ \textbf{in}\ t \longrightarrow \Delta \vdash \textbf{let rec}\ x\ = abs'\ \textbf{in}\ t}\ \textsc{E\_LetRec1}$$

$$\frac{}{\Delta \vdash \textbf{let rec}\ x\ = valAbs\ \textbf{in}\ t \longrightarrow \Delta, \{x \leftarrow \textbf{let rec}\ x\ = valAbs\ \textbf{in}\ valAbs\} \vdash t}\ \textsc{E\_LetRec2}$$

$$\frac{}{\Delta \vdash \textbf{match failure}\ exnval\ \textbf{with}\ p_1 \to t_1 | .. | p_n \to t_n\ \textbf{end} \longrightarrow \Delta \vdash \textbf{failure}\ exnval}\ \textsc{E\_MatchFailure}$$

$$\frac{\Delta \vdash t \longrightarrow \Delta \vdash t'}{\Delta \vdash \textbf{match}\ t\ \textbf{with}\ p_1 \to t_1 | .. | p_n \to t_n\ \textbf{end} \longrightarrow \Delta \vdash \textbf{match}\ t'\ \textbf{with}\ p_1 \to t_1 | .. | p_n \to t_n\ \textbf{end}}\ \textsc{E\_Match}$$

$$\frac{v\ \textbf{matches}\ p_1 \rhd \Delta'}{\Delta \vdash \textbf{match}\ v\ \textbf{with}\ p_1 \to t_1 | .. | p_n \to t_n\ \textbf{end} \longrightarrow \Delta \cup \Delta' \vdash t_1}\ \textsc{E\_MatchFound}$$

$$\frac{\textbf{not}\ (v\ \textbf{matches}\ p_1 \rhd \Delta')}{\Delta \vdash \textbf{match}\ v\ \textbf{with}\ p_1 \to t_1 | p_2 \to t_2 | .. | p_n \to t_n\ \textbf{end} \longrightarrow \Delta \vdash \textbf{match}\ v\ \textbf{with}\ p_2 \to t_2 | .. | p_n \to t_n\ \textbf{end}}\ \textsc{E\_MatchS}$$

$$\frac{\Delta \vdash t \longrightarrow \Delta \vdash t'}{\Delta \vdash (t : [[\mathit{eff}]]\,T) \longrightarrow \Delta \vdash t'}\ \textsc{E\_Annot}$$

$$\frac{\Delta \vdash t \longrightarrow \Delta \vdash t'}{\Delta \vdash \textbf{fail}\,[T]Exc\ v_1\ ..\ v_n\ t\ t_1\ ..\ t_n \longrightarrow \Delta \vdash \textbf{fail}\,[T]Exc\ v_1\ ..\ v_n\ t'\ t_1\ ..\ t_n}\ \textsc{E\_FailureArgs}$$

$$\frac{}{\Delta \vdash \textbf{fail}\,[T]\,Exc\ v_1\ ..\ v_n \longrightarrow \Delta \vdash \textbf{failure}\ Exc\ v_1\ ..\ v_n}\ \text{E\_Failure}$$

$$\frac{\Delta \vdash t \longrightarrow \Delta \vdash t'}{\Delta \vdash \textbf{try}\ t\ \textbf{with}\ pe_1 \rightarrow t_1 |\ ..\ |pe_n \rightarrow t_n\ \textbf{end} \longrightarrow \Delta \vdash \textbf{try}\ t'\ \textbf{with}\ pe_1 \rightarrow t_1 |\ ..\ |pe_n \rightarrow t_n\ \textbf{end}}\ \text{E\_Try}$$

$$\frac{}{\Delta \vdash \textbf{try}\ v\ \textbf{with}\ pe_1 \rightarrow t_1 |\ ..\ |pe_n \rightarrow t_n\ \textbf{end} \longrightarrow \Delta \vdash v}\ \text{E\_TryNoFailure}$$

$$\frac{\textbf{not}\ (exnval\ \textbf{matches}\ pe_1 \rhd \Delta')}{\Delta \vdash \textbf{try failure}\ exnval\ \textbf{with}\ pe_1 \rightarrow t_1\ \textbf{end} \longrightarrow \Delta \vdash \textbf{failure}\ exnval}\ \text{E\_TryNotFound}$$

$$\frac{exnval\ \textbf{matches}\ pe_1 \rhd \Delta'}{\Delta \vdash \textbf{try failure}\ exnval\ \textbf{with}\ pe_1 \rightarrow t_1 |\ ..\ |pe_n \rightarrow t_n\ \textbf{end} \longrightarrow \Delta \cup \Delta' \vdash t_1}\ \text{E\_TryFound}$$

$$\frac{\textbf{not}\ (exnval\ \textbf{matches}\ pe_1 \rhd \Delta')}{\Delta \vdash \textbf{try failure}\ exnval\ \textbf{with}\ pe_1 \rightarrow t_1 | pe_2 \rightarrow t_2 |\ ..\ |pe_n \rightarrow t_n\ \textbf{end} \longrightarrow \Delta \vdash \textbf{try failure}\ exnval\ \textbf{with}\ pe_2 \rightarrow t_2 |\ ..\ |pe_n}$$

$\boxed{exnval\ \textbf{matches}\ pe \rhd \Delta}$ Exception pattern matching with substitution creation

$$\frac{}{Exc\ v_1\ ..\ v_n\ \textbf{matches}\ Exc\ x_1\ ..\ x_n \rhd \emptyset, \{x_1 \leftarrow v_1\ ..\ x_n \leftarrow v_n\}}\ \text{ExnMatches\_Matches}$$

$\boxed{v\ \textbf{matches}\ p \rhd \Delta}$ Pattern matching with substitution creation

$$\frac{}{v\ \textbf{matches}\ x \rhd \emptyset, \{x \leftarrow v\}}\ \text{Matches\_Any}$$

$$\frac{v_1\ \textbf{matches}\ p_1 \rhd \Delta_1 \quad .. \quad v_n\ \textbf{matches}\ p_n \rhd \Delta_n}{\textbf{TConstr}\ V\ v_1\ ..\ v_n\ \textbf{matches}\ V\ p_1\ ..\ p_n \rhd \Delta_1 \cup\ ..\ \cup \Delta_n}\ \text{Matches\_Matches}$$

```
Definition rules:         103 good    0 bad
Definition rule clauses:  224 good    0 bad
```