Hazel PHI: 10-modules

how to read

800000	kinds	800080	signatures
008000	types (constructors)	008080	modules
000080	terms		

notes

$$\begin{tikzcd}
 & \kappa & & \hat{sig} \arrow[r, rightsquigarrow] & sig \\
\hat{\tau} \arrow[rr, rightsquigarrow] & & \tau & \hat{mod} \arrow[r, rightsquigarrow] & mod \\
\hat{\delta} \arrow[rr, rightsquigarrow] & & \delta & &
\end{tikzcd}$$

syntax

kind	κ	$::=$	$\texttt{Type}$	kind of types
		$\mid$	$\texttt{S}(\tau)$	singleton kind
		$\mid$	$\texttt{KHole}$	kind hole
		$\mid$	$\Pi_{t::\kappa_1}.\kappa_2$	dependent function kind
		$\mid$	$\Sigma_{t::\kappa_1}.\kappa_2$	dependent product kind

external HTyp	$\hat{\tau}$	$::=$	t	type variable
		$\mid$	bse	base type
		$\mid$	$\hat{\tau_1} \oplus \hat{\tau_2}$	type binop
		$\mid$	$[\hat{\tau}]$	list type
		$\mid$	$\lambda t :: \kappa.\hat{\tau}$	type function
		$\mid$	$\hat{\tau_1}\ \hat{\tau_2}$	type application
		$\mid$	$\langle \hat{\tau_1}, \hat{\tau_2} \rangle$	type pair
		$\mid$	$\pi_1\ \hat{\tau}$	type projection
		$\mid$	$\pi_2\ \hat{\tau}$	type projection
		$\mid$	$\{lab_1 \hookrightarrow \hat{\tau_1}, \ldots\ lab_n \hookrightarrow \hat{\tau_n}\}$	labelled product type (record)
		$\mid$	$mod.lab$	module type projection
		$\mid$	$(\!\|\|\!)$	empty type hole
		$\mid$	$(\!\|\hat{\tau}\|\!)$	nonempty type hole

internal HTyp	τ	$::=$	t	type variable
		$\mid$	bse	base type
		$\mid$	$\tau_1 \oplus \tau_2$	type binop
		$\mid$	$[\tau]$	list type
		$\mid$	$\lambda t :: \kappa . \tau$	type function
		$\mid$	$\tau_1 \ \tau_2$	type application
		$\mid$	$\langle \tau_1 , \tau_2 \rangle$	type pair
		$\mid$	$\pi_1 \ \tau$	type projection
		$\mid$	$\pi_2 \ \tau$	type projection
		$\mid$	$\{ lab_1 \hookrightarrow \tau_1 , \ldots \ lab_n \hookrightarrow \tau_n \}$	labelled product type (record)
		$\mid$	$mod.lab$	module type projection
		$\mid$	$(\!\|\!)$	empty type hole
		$\mid$	$(\!\|\tau\|\!)$	nonempty type hole
base type	bse	$::=$	`Int`	
		$\mid$	`Float`	
		$\mid$	`Bool`	
HTyp BinOp	$\oplus$	$::=$	$\times$	
		$\mid$	$+$	
		$\mid$	$\rightarrow$	
external expression	$\hat{\delta}$	$::=$	x	
		$\mid$	$\texttt{signature}\ s = sig\ \texttt{in}\ \hat{\delta}$	
		$\mid$	$\texttt{module}\ m = mod\ \texttt{in}\ \hat{\delta}$	
		$\mid$	$\texttt{module}\ m :: s = mod\ \texttt{in}\ \hat{\delta}$	
		$\mid$	$mod.lab$	module term projection
		$\mid$	*elided*	
internal expression	δ	$::=$	x	
		$\mid$	$\texttt{signature}\ s = sig\ \texttt{in}\ \delta$	
		$\mid$	$\texttt{module}\ m :: s = mod\ \texttt{in}\ \delta$	
		$\mid$	$mod.lab$	module term projection
		$\mid$	*elided*	

external signature	$\hat{sig}$	$::=$	$\hat{s}$	signature variable
		$\vert$	$\{ sdecs \}$	structure signature
		$\vert$	$\Pi_{m::\hat{sig}_1}.\hat{sig}_2$	functor signature
internal signature	sig	$::=$	s	signature variable
		$\vert$	$\{ sdecs \}$	structure signature
		$\vert$	$\Pi_{m::sig_1}.sig_2$	functor signature
external module	$\hat{mod}$	$::=$	m	module variable
		$\vert$	$\{ sbnds \}$	structure
		$\vert$	$\lambda m :: sig.\hat{mod}$	functor
		$\vert$	$\hat{mod}_1\ \hat{mod}_2$	functor application
		$\vert$	$\hat{mod}.lab$	submodule projection
internal module	mod	$::=$	m	module variable
		$\vert$	$\{ sbnds \}$	structure
		$\vert$	$\lambda m :: sig.mod$	functor
		$\vert$	$mod_1\ mod_2$	functor application
		$\vert$	$mod.lab$	submodule projection
signature declarations	$sdecs$	$::=$	ϵ	
		$\vert$	$sdec, sdecs$	
signature declaration	$sdec$	$::=$	`type` lab	
		$\vert$	`type` $lab = \tau$	
		$\vert$	`val` $lab : \tau$	
		$\vert$	`module` $lab :: sig$	
structure bindings	$sbnds$	$::=$	ϵ	
		$\vert$	$sbnd, sbnds$	
structure binding	$sbnd$	$::=$	`type` $t = \tau$	
		$\vert$	`let` $x : \tau = \delta$	
		$\vert$	`module` $m = mod$	
		$\vert$	`module` $m :: s = mod$	

statics

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

`KCSubsumption`

$$\frac{test}{test}$$