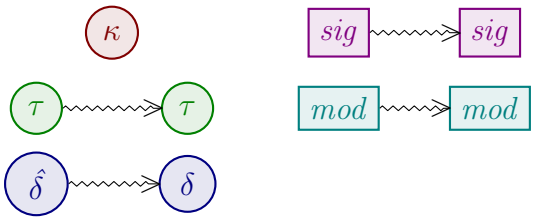
Hazel PHI: 10-modules

how to read

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

notes



syntax

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

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

external expression	$\hat{\delta}$	::=	x
		|	`signature` $s = sig$ `in` $\hat{\delta}$
		|	`module` $m = mod$ `in` $\hat{\delta}$
		|	`module` $m :: s = mod$ `in` $\hat{\delta}$
		|	$mod.lab$ module term projection
		|	*elided*
internal expression	δ	::=	x
		|	`signature` $s = sig$ `in` δ
		|	`module` $m :: s = mod$ `in` δ
		|	$mod.lab$ module term projection
		|	*elided*

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

statics

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

$$\texttt{KCSubsumption}\ \frac{test}{test}$$