

# Hazel PHI: 10-modules

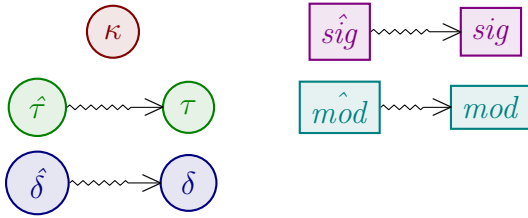
## how to read

---

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

## notes

---



## syntax

---

kind $\kappa$	::=	<b>Type</b>	kind of types
		$S(\tau)$	singleton kind
		<b>KHole</b>	kind hole
		$\Pi_{t::\kappa_1}.\kappa_2$	dependent function kind
		$\Sigma_{t::\kappa_1}.\kappa_2$	dependent product kind
external HType $\hat{\tau}$	::=	$t$	type variable
		$bse$	base type
		$\hat{\tau}_1 \oplus \hat{\tau}_2$	type binop
		$[\hat{\tau}]$	list type
		$\lambda t :: \kappa. \hat{\tau}$	type function
		$\hat{\tau}_1 \hat{\tau}_2$	type application
		$\langle \hat{\tau}_1, \hat{\tau}_2 \rangle$	type pair
		$\pi_1 \hat{\tau}$	type projection
		$\pi_2 \hat{\tau}$	type projection
		$\{lab_1 \hookrightarrow \hat{\tau}_1, \dots lab_n \hookrightarrow \hat{\tau}_n\}$	labelled product type (record)
		$mod.lab$	module type projection
		$\langle \rangle$	empty type hole
		$\langle \hat{\tau} \rangle$	nonempty type hole

internal HType	$\tau ::=$	$t$	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
		$\{lab_1 \hookrightarrow \tau_1, \dots lab_n \hookrightarrow \tau_n\}$	labelled product type (record)
		$mod.lab$	module type projection
		$\langle \rangle$	empty type hole
		$\langle \tau \rangle$	nonempty type hole

base type	$bse ::=$	<code>Int</code>
		<code>Float</code>
		<code>Bool</code>

HType BinOp	$\oplus ::=$	$\times$
		$+$
		$\rightarrow$

external expression	$\hat{\delta} ::=$	$x$	
		<code>signature</code> $s = sig$ <code>in</code> $\hat{\delta}$	
		<code>module</code> $m = mod$ <code>in</code> $\hat{\delta}$	
		<code>module</code> $m :: s = mod$ <code>in</code> $\hat{\delta}$	
		$mod.lab$	module term projection
		<i>elided</i>	

internal expression	$\delta ::=$	$x$	
		<code>signature</code> $s = sig$ <code>in</code> $\delta$	
		<code>module</code> $m :: s = mod$ <code>in</code> $\delta$	
		$mod.lab$	module term projection
		<i>elided</i>	

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

## statics

---

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

KCSubsumption

$\frac{test}{test}$