n, k, i, j, q, m	Index variables for meta-lists
num, $numZero$, $numOne$	Numeric literals
nat	Natural numbers
hex	Bit vector literal, specified by C-style hex number
$binary$	Bit vector literal, specified by C-style binary number
$regexp$	Regular expresions, as a string literal
id	Identifiers
$string\_val$	String literals
$real\_val$	Real number literal
$tvar$	Base-type variable
$ctor$	Constructor
$field$	Record Field
p	Projection index
pos	Source file position
u	Mutable Variables

lit	::=			Literal constant
	|	()		
	|	**bitzero**		
	|	**bitone**		
	|	**true**		
	|	**false**		
	|	*num*		natural number constant
	|	*string_val*		string constant
	|	**undefined**		undefined-value constant
	|	*real_val*		
order	::=			Vector order specification
	|	**inc**		
	|	**dec**		
	|	**def_order**		default order
loop	::=			Loop variant
	|	**while**		
	|	**until**		
$x,\ z,\ y,\ kp,\ fp$	::=			Immutable variables incl. function names
	|	*id*		general variable
	|	z		value variable bound in refinement type
σ	::=			Substitution for base types
	|	**empty**		
	|	$\sigma, b^p / tvar$		
b^p	::=			Base Type
	|	*tvar*		type variable
	|	**tid** *id*		type identifier
	|	**int**		
	|	**bool**		
	|	**bit**		
	|	**unit**		
	|	**real**		
	|	**vec** $order\ b^p$		vector (length is part of constraint)
	|	**list** b^p		list
	|	$(b^p_1, .., b^p_n)$		tuple
	|	$id\langle ctor_1 : \tau^p_1, \ldots, ctor_n : \tau^p_n\rangle$		union
	|	$\{field_1 : b^p_1, \ldots, field_n : b^p_n\}$		record
	|	**undef**		
	|	**reg t**		register
	|	**string**		
	|	$b^p_1[b^p_2/tvar]$	M	
	|	$b^p_1[\sigma]$	M	
	|	**exception**		

	$\mid$	$\{num_1, .., num_n\}$		finite set of integers
τ^p	::=			Refinement type
	$\mid$	$\{z : b^p \mid \phi^p\}$		
	$\mid$	$\tau^p[v^p/z]$	M	substitution of value variables
	$\mid$	$\tau^p[b^p/z]$	M	substitition of base-type variables
a^p	::=			Dependent Function and Monotype Type. Unlike Sail we distinguish in
	$\mid$	τ^p		
	$\mid$	$x : b^p[\phi^p] - > \tau^p$		
bop	::=			Binary operators
	$\mid$	$+$		
	$\mid$	$-$		
	$\mid$	$*$		
	$\mid$	**div**		
	$\mid$	**mod**		
	$\mid$	$\leq$		
	$\mid$	$<$		
	$\mid$	$>$		
	$\mid$	$\geq$		
	$\mid$	$=$		
	$\mid$	&		
	$\mid$	$\mid\mid$		
	$\mid$	$\neq$		
uop	::=			Unary operators
	$\mid$	**len**		vector length
	$\mid$	**exp**		base 2 exponent
	$\mid$	**neg**		negation
	$\mid$	**not**		boolean not
ce^p	::=			Constraint expression
	$\mid$	v^p		
	$\mid$	$ce_1^p \, bop \, ce_2^p$		
	$\mid$	$\textbf{sum}\,(ce_1^p \ldots ce_n^p)$		
	$\mid$	$uop \, ce^p$		
	$\mid$	$\pi_i ce^p$		tuple projection
	$\mid$	$x.field$		field access
	$\mid$	(ce^p)	S	
	$\mid$	$ce^p[v^p/x]$	M	
ϕ^p	::=			Refinement Constraints - Quantifier free logic of uninterpreted function
	$\mid$	$\top$		
	$\mid$	$\perp$		
	$\mid$	$\phi_1^p \wedge \phi_2^p$		
	$\mid$	$\phi_1^p \wedge \, .. \, \wedge \phi_n^p$		

	|	$\phi_1^p \vee \phi_2^p$		
	|	$\neg \phi^p$		
	|	$ce_1^p = ce_2^p$		
	|	$ce_1^p \leq ce_2^p$		
	|	(ϕ^p)	S	
	|	$\phi^p[v^p/x]$	M	
	|	$\phi^p[ce_1^p/x_1 \, .. \, ce_n^p/x_n]$	M	
	|	$\phi_1^p \Longrightarrow \phi_2^p$		
v^p	::=			Values
	|	lit		
	|	x		
	|	$[v_1^p, \, ... \, , v_n^p]$		vector
	|	$[\|v_1^p, \, .. \, , v_n^p\|]$		
	|	$\mathbf{cons}\, v_1^p\, v_2^p$		list cons
	|	$ctor\, v^p$		union constructor
	|	$\{field_1 = v_1^p, \, ... \, , field_n = v_n^p\}$		record
	|	$(v_1^p, \, ... \, , v_n^p)$		tuple
	|	$v_1^p[v_2^p/x]$	M	substitution
	|	(v^p)	S	
	|	$\mathbf{proj}\, p\, v^p$		tuple projection
loc	::=			Source file location
	|	**unknown**		
	|	$\mathbf{range}\, pos_1\, pos_2$		
γ^p	::=			List of $x : b[\phi]$ triples
	|	ϵ	S	
	|	$\overline{kp_i : b_i^p[\phi_i^p]}^{\,i}$	S	
	|	$(\gamma_1^p \, .. \, \gamma_n^p)$	S	
	|	(γ_1^p, γ_2^p)	S	
	|	$x : b^p[\phi^p], \gamma^p$		
	|	$x : b^p[\phi^p]$		
pat^p	::=			Patterns
	|	lit		
	|	$\_$		
	|	$(pat^p\, \mathbf{as}\, x)$		pattern as an immutable variable
	|	$(\tau^p) pat^p$		pattern cast
	|	id		
	|	$pat^p\, \mathbf{as}\, \tau^p$		pattern as a type-variable
	|	$id(pat_1^p, \, ... \, , pat_n^p)$		
	|	$[pat_1^p, \, .. \, , pat_n^p]$		vector pattern
	|	$pat_1^p : \, ... \, : pat_n^p$		concatenated vector pattern
	|	$(pat_1^p, \, .. \, , pat_n^p)$		tuple pattern
	|	$[\|\|pat_1^p, \, .. \, , pat_n^p\|\|]$		list pattern
	|	(pat^p)	S	

	|	$pat_1^p :: pat_2^p$		cons pattern
	|	pat_1^p ^^ ... ^^pat_n^p		string append pattern
	|	$pat^p[v^p/x]$	M	
$pexp^p$	::=			Pattern Expressions
	|	$pat^p \Rightarrow e^p$		
	|	pat^p **when** $e_1^p \Rightarrow e_2^p$		guarded pattern expression
	|	$pexp^p[v^p/x]$	M	
$letbind$	::=			
	|	**let** $pat^p = e^p$		let, implicit type (pat^p must be total)
$lexp^p$	::=			L-value expressions. Subset of Sail's but
	|	u		mutable variable
	|	$(\tau^p)u$		cast
	|	$(lexp_0^p, .., lexp_n^p)$		multiple (non-memory) assignment
	|	$lexp^p.id$		struct field
e^p	::=			Expressions
	|	v^p		value
	|	u		mutable variable
	|	$[e_1^p; ..; e_n^p]$		vector concatenation
	|	$(e_1^p, .., e_n^p)$		tuple
	|	$fp\, e^p$		function application
	|	$bop\, e_1^p\, e_2^p$		binary operation
	|	$uop\, e^p$		unary operation
	|	$\pi_1 e^p$		projection from a tuple
	|	$ctor\, e^p$		constructor application
	|	$e^p.field$		field access
	|	$e^p[v^p/x]$	M	Value variable substitution
	|	(e^p)	S	
	|	**sizeof** ce^p		progmram level expression from t. No
	|	**cast** $\tau^p\, e^p$		type cast
	|	$\{field_0 = e_0^p, \ldots, field_n = e_n^p\}$		record
	|	$\{e^p$ **with** $field_0 = e_0^p; \ldots; field_n = e_n^p\}$		record update
	|	$letbind$ **in** e_2^p		let binding
	|	**let** $x : \tau^p = e_1^p$ **in** e_2^p	bind x in e_2^p	let binding with type annotation
	|	**if** e_1^p **then** e_2^p **else** e_3^p		if statement
	|	$\{e_1^p; \ldots; e_n^p\}$		sequential block
	|	**match** $e^p\{pexp_1^p, \ldots, pexp_n^p\}$		case/match statement
	|	$lexp^p := e_1^p$ **in** e_2^p		mutable variable assignment and scopi
	|	**exit** e^p		program exit
	|	**return** e^p		function return
	|	**throw** e^p		exception throw
	|	**try** e^p **catch** $pexp_1^p .. pexp_n^p$		try block
	|	**constraint** ϕ^p		constraint
	|	$loop\, e_1^p\, e_2^p$		general loop

	$\mid$	$\textbf{while}\, e_1^p\, \textbf{do}\, e_2^p$	S	while loop
	$\mid$	$\textbf{repeat}\, e_1^p\, \textbf{until}\, e_2^p$	S	repeat loop
	$\mid$	$\textbf{for}\,(id\,\textbf{from}\, e_1^p\,\textbf{to}\, e_2^p\,\textbf{by}\, e_3^p\,\textbf{in}\, order)e_4^p$		
	$\mid$	$\textbf{assert}\, e_1^p\, e_2^p$		assert
	$\mid$	$[e_1^p, \ldots, e_n^p]$		vector
	$\mid$	$[\|\|e_1^p, .., e_n^p\|\|]$		list
	$\mid$	$e_1^p :: e_2^p$		list cons
$funcl$	$::=$			Function clause
	$\mid$	$id\; pexp^p$		
$tannot\_opt$	$::=$			Function type annotation
	$\mid$			
	$\mid$	$\gamma^p\, \tau^p$		
	$\mid$	a^p		
$scattered\_def$	$::=$			Scattered definition
	$\mid$	$\textbf{scattered function}\; tannot\_opt\; id$		
	$\mid$	$\textbf{scattered union}\; id\; \overline{kp_i : b_i^p[\phi_i^p]}^{\,i}$		
	$\mid$	$\textbf{union clause}\; id_1 = id_2 : \tau^p$		
	$\mid$	$\textbf{function clause}\; funcl$		
	$\mid$	$\textbf{end}\; id$		
def^p	$::=$			Definitions
	$\mid$	$\textbf{function}\; a^p\; funcl_1\; \textbf{and} \ldots \textbf{and}\; funcl_n$		function
	$\mid$	$\textbf{typedef}\; id = \forall\, \overline{kp_i : b_i^p[\phi_i^p]}^{\,i}\; \tau^p$		type
	$\mid$	$\textbf{val}\; id : a^p$		val type spec
	$\mid$	$letbind$		let binding
	$\mid$	$\textbf{register}\; \tau^p\; x$		register
	$\mid$	$\textbf{overload}\; id[id_1; ..; id_n]$		operator overloading
	$\mid$	$scattered\_def$		scattered definition
	$\mid$	$\textbf{default}\; order$		default order spec
$progp$	$::=$			program
	$\mid$	$\mathrm{def}_1^p .. \mathrm{def}_n^p$		program is just a list of definitions
$terminals$	$::=$			
	$\mid$	$\vdash_v$		
	$\mid$	$\vdash$		
	$\mid$	$\dashv$		
	$\mid$	$\models$		
	$\mid$	$\Leftarrow$		
	$\mid$	$\Rightarrow$		
	$\mid$	$\vee$		
	$\mid$	$\wedge$		
	$\mid$	$\lessapprox$		
	$\mid$	$\exists$		

	|	$\Longrightarrow$	
	|	$\longrightarrow$	
	|	$\rightsquigarrow$	
	|	$\rightsquigarrow$	
	|	$\langle$	
	|	$\rangle$	
	|	$\exists$	
	|	$\forall$	
	|	$\in$	
	|	$\in$	
	|	$\notin$	
	|	$\vdash$	
	|	$\subseteq$	
	|	^^	
Θ	::=		Type definition context
	|	ϵ	
	|	Θ, def^p	
	|	$\Theta, id : \forall\, \overline{kp_i : b_i^p[\phi_i^p]}^{\,i}\ \tau^p$	
	|	$\Theta, order$	
Φ	::=		Function context
	|	ϵ	empty context
	|	$\Phi, id : a^p$	add a function definition
	|	$\Phi, id[id_1 \mathinner{..} id_n]$	add overload spec
Γ	::=		Immutable variable context
	|	ϵ	empty context
	|	$\Gamma, x : b^p[\phi^p]$	add immutable variable
	|	(Γ)	S
	|	Γ_1, Γ_2	append
	|	$\Gamma_1, \mathinner{..}, \Gamma_n$	append many
	|	$\Gamma, \gamma^p, x : b^p[\phi^p]$	add list of immutable variables and single one
	|	$\Gamma, \gamma_1^p, \mathinner{..}, \gamma_n^p$	add many immutable variables
Δ	::=		Mutable variable context
	|	ϵ	empty context
	|	Δ_1, Δ_2	append
	|	(Δ)	S
	|	$\Delta + u : \tau^p$	add mutable variable
	|	$\Delta + (u_1 : \tau_1^p, \mathinner{..}, u_n : \tau_n^p)$	add list of mutable variables
	|	$\Delta + +u : \tau^p$	update mutable variable
xlist	::=		Mutable variable lists
	|	ϵ	
	|	$x_1, \mathinner{..}, x_n$	
	|	$[xlist_1, \mathinner{..}, xlist_n]$	

$formula$ ::=

| $judgement$
| $formula_1 \quad .. \quad formula_n$
| $[\Gamma] \Rightarrow \phi^p valid$
| $\Theta; \Gamma \models \phi^p$
| $x : b^p[\phi^p] \in \Gamma$
| $x \notin \Gamma$
| $u \notin \Delta$
| $u : \tau^p \in \Delta$
| $id : a^p \notin \Phi$
| $x \in \mathrm{dom}(\Gamma)$
| $\mathbf{f} \in \mathrm{dom}(\Phi)$
| $p(\mathbf{f}) = e^p$
| $\mathrm{vars}(\phi^p) \subseteq \mathrm{dom}(\Gamma) \cup \{x\}$
| $\mathbf{f} : (x : b^p[\phi^p]) : \tau^p = e^p \in \Phi$
| $\mathbf{l} = \{z : b^p | \phi^p\}$
| $\tau^p = \mathbf{lookup}\, \dot{ctor} \langle \dot{ctor}_1 : \tau_1^p, .., \dot{ctor}_n : \tau_n^p \rangle$
| $\Theta(\dot{ctor} \mapsto (\tau_1^p, \tau_2^p))$ Returns union type ar
| $\Theta(id \mapsto \tau_1^p, \tau_2^p)$ Returns union type ar
| $\Theta(\dot{ctor} \mapsto b^p \tau^p)$
| $x_1 : b_1^p[\phi_1^p], .., x_n : b_n^p[\phi_n^p] \ \in \ \Gamma$
| $\mathbf{default}\, order \in \Theta$
| $\tau^p = \mathbf{lookup\_return}\, \Gamma$
| $\Delta = \mathbf{add\_mvar}\, \Gamma\, x\, \tau^p$
| $\Gamma_1 = \mathbf{add\_return}\, \Gamma_2\, \tau^p$
| $b^p\, \tau^p = \mathbf{lookup\_field\_record\_type}\, \Theta\, id$
| $\tau_1^p\, \tau_2^p = \mathbf{lookup\_field\_and\_record\_type}\, \Theta\, u$
| $b^p\, \tau_2^p = \mathbf{lookup\_field\_record\_type}\, \Theta\, field$
| $\tau^p = \mathbf{lookup\_fields}\, \Theta\, field_1 \,..\, field_n$
| $b_1^p \,..\, b_n^p = \mathbf{lookup\_types\_for}\, b^p\, field_1 \,..\, field_m$
| $\Phi' = \mathbf{add\_fun}\, \Phi\, x\, b^p\, \phi^p\, \tau_2^p\, funcl_1 \,...\, funcl_n$
| $\Gamma_1 = \Gamma_2, x : b^p[\phi^p]$
| $a_1^p, .., a_n^p = \mathbf{lookup\_fun\_type}\, \Theta\, \Phi\, fp$
| $\Gamma' = \Gamma, \Delta$
| $x_1 \,..\, x_n \ \notin \ \Gamma$
| $e_1^p + e_2^p = v^p$
| $e_1^p \leq e_2^p = v^p$
| $x1 = e_2^p$
| $e_1^p \neq e_2^p$
| $x \# e^p$
| $\phi_1^p \,..\, \phi_n^p = \mathbf{constraints}\, \tau_1^p \,..\, \tau_m^p$
| $\phi_1^p \,..\, \phi_n^p = \mathbf{constraints\_with\_proj}\, \tau_1^p \,..\, \tau_m^p$
| $\phi_1^p \,..\, \phi_n^p = \mathbf{constraints\_with\_field\_proj}\, field_1 : \tau_1^p \,..\, field_m : \tau_m^p$
| $Z2, (id_1 \,..\, id_n) = \mathbf{fresh}\, Z1$ Construct locally fresh
| $Z2, id = \mathbf{fresh\_for\_tuple}\, Z1 \tau_1^p \,..\, \tau_n^p$ Construct globally fre
| $Z2, id = \mathbf{fresh\_for\_record}\, Z1 field_1 \,..\, field_n$ Construct globally fre

	|	**unsatisfiable** $=$ **check_sat Z**	
	|	$x, \Gamma' =$ **fresh** Γ	
	|	$x =$ **fresh** Γ	
	|	$\phi^p =$ **proj_c_conj** $\phi^p_1 \, .. \, \phi^p_n$	Replaces all zi in ci with
	|	$\sigma = (b^p_1 \; b^p_2)$	
	|	$\{ num_1, \, .., \, num_n \} \subseteq \{ num'_1, \, .., \, num'_m \}$	
	|	$num =$ **list_lepn** $[v^p_1, \, .., \, v^p_n]$	
	|	$x_1 \, .. \, x_m, \gamma^p =$ **mk_proj_vars** $x \, b^p_1, \, .., \, b^p_n$	
	|	**SATIS** Γ	
	|	$b^p =$ **single_base** $b^p_1 \, .. \, b^p_n$	
	|	$v^p =$ **mk_ctor_v** $id \; xlist$	
	|	$x : b^p[\phi^p] -> \{ \#0 : b^p_2 \mid \phi^p_2 \} =$ **match_arg** $\tau^p_1(a^p_1, \, .., \, a^p_n)$	
	|	$\sigma =$ **UNIFY** $b^p_1 \, b^p_2$	
	|	$b^p_1 \, .. \, b^p_n =$ **b_of** $(\tau^p_1 \, .. \, \tau^p_m)$	Extract bases from types
subtyping	::=		
	|	$\Theta \vdash b^p_1 \lesssim b^p_2$	Subtyping of base types
	|	$\Theta; \Gamma \vdash \tau^p_1 \lesssim \tau^p_2$	Subtyping
typing_v	::=		
	|	$\Theta; \Gamma \vdash_v v^p \Rightarrow \tau^p$	Type synthesis for values
	|	$\Theta; \Gamma \vdash_v v^p \Leftarrow \tau^p$	Type check v^p is τ^p wher
typing_pat	::=		
	|	$\Theta; \Gamma \vdash pat^p \Rightarrow \tau^p \rightsquigarrow x; \gamma^p; xlist$	Infer type of patmtern *pa*
	|	$\Theta; \Gamma \vdash pat^p_1 \, .. \, pat^p_n \Leftarrow b^p_1, \, .., \, b^p_m \rightsquigarrow \gamma^p; x_1 \, .. \, x_j$	Type check of list of patt
	|	$\Theta; \Gamma \vdash pat^p \Leftarrow \tau^p \rightsquigarrow x; \gamma^p; xlist$	Type check pattern pat^p
typing_lexp	::=		
	|	$\Theta; \Gamma; \Delta \vdash lexp^p \Rightarrow \tau^p \rightsquigarrow \Delta'$	Type synthesis for l-value
	|	$\Theta; \Gamma; \Delta \vdash (lexp^p_1 \, .. \, lexp^p_n) \Rightarrow (\tau^p_1 \, .. \, \tau^p_m) \rightsquigarrow \Delta'$	Type synthesis for a list c
	|	$\Theta; \Gamma; \Delta \vdash (lexp^p_1 \, .. \, lexp^p_n) \Leftarrow (b^p_1 \, .. \, b^p_m) \rightsquigarrow \Delta'$	Type check for a list l-val
	|	$\Theta; \Gamma; \Delta \vdash lexp^p \Leftarrow \tau^p \rightsquigarrow \Delta'$	Type check l-value expres
typing_e	::=		
	|	$\Theta; \Phi; \Gamma; \Delta \vdash pexp^p \Leftarrow \tau^p_1, \tau^p_2 \rightsquigarrow \Gamma'$	Type check pattern is τ^p_1
	|	**match_overload** $\Theta \, \Gamma \{ z_2 : b^p_2 \mid \phi^p_2 \} (a^p_1, \, ..., \, a^p_n)(a^p)\sigma$	Find list of functions hav
	|	$\Theta; \Phi; \Gamma; \Delta \vdash (a^p_1, \, ..., \, a^p_n) e^p \Rightarrow \tau^p \rightsquigarrow x; \gamma^p$	Type synthesis for overlo
	|	$\Theta; \Phi; \Gamma; \Delta \vdash e^p_1 \, .. \, e^p_m \Rightarrow \tau^p_1 \, .. \, \tau^p_n \rightsquigarrow x_1 \, .. \, x_j; \gamma^p$	Type synthesis for expres
	|	$\Theta; \Phi; \Gamma; \Delta \vdash letbind \rightsquigarrow \gamma^p$	Bindings γ for a let-bind
	|	$\Theta; \Phi; \Gamma \vdash_e e^p \Rightarrow \tau^p \rightsquigarrow x; \gamma^p$	Infer that type of e^p is τ^p
	|	$\Theta; \Phi; \Gamma; \Delta \vdash lexp^p = e^p \rightsquigarrow \Delta'; \gamma^p$	Assignment expression ty
	|	$\Theta; \Phi; \Gamma; \Delta \vdash e^p_1 \, .. \, e^p_n \Leftarrow \tau^p_1 \, .. \, \tau^p_m \rightsquigarrow \Gamma'$	Type check list of express
	|	$\Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau^p$	Type check of e^p against
def_checking	::=		
	|	$\Theta; \Phi; \Gamma \vdash funcl_1 \, .. \, funcl_n \Leftarrow x : b^p[\phi^p], \tau^p_2 \rightsquigarrow \Phi'; \Gamma'$	

	|	$\Theta; \Phi; \Gamma \vdash \mathrm{def}^p \leadsto \Phi'; \Gamma'$
	|	$\Theta; \Phi; \Gamma \vdash \mathrm{def}_1^p \ldots \mathrm{def}_n^p \leadsto \Theta'; \Phi'; \Gamma'$
judgement	::=	
	|	*subtyping*
	|	*typing_v*
	|	*typing_pat*
	|	*typing_lexp*
	|	*typing_e*
	|	*def_checking*
user_syntax	::=	
	|	n
	|	*num*
	|	*nat*
	|	*hex*
	|	*binary*
	|	*regexp*
	|	*id*
	|	*string_val*
	|	*real_val*
	|	*tvar*
	|	*ctor*
	|	*field*
	|	p
	|	*pos*
	|	u
	|	*lit*
	|	*order*
	|	*loop*
	|	x
	|	σ
	|	b^p
	|	τ^p
	|	a^p
	|	*bop*
	|	*uop*
	|	ce^p
	|	ϕ^p
	|	v^p
	|	*loc*
	|	γ^p
	|	pat^p
	|	$pexp^p$
	|	*letbind*
	|	$lexp^p$

$$
\begin{array}{ll}
| & e^p \\
| & funcl \\
| & tannot\_opt \\
| & scattered\_def \\
| & \mathrm{def}^p \\
| & progp \\
| & terminals \\
| & \Theta \\
| & \Phi \\
| & \Gamma \\
| & \Delta \\
| & xlist \\
| & formula
\end{array}
$$

$\boxed{\Theta \vdash b_1^p \lesssim b_2^p}$ Subtyping of base types

$$\frac{\sigma = (b_1^p\ b_2^p)}{\Theta \vdash b_1^p \lesssim b_2^p} \quad \text{SUBTYPE\_BASE\_REFL}$$

$$\frac{\{num_1, \ldots, num_n\} \subseteq \{num_1', \ldots, num_m'\}}{\Theta \vdash \{num_1, \ldots, num_n\} \lesssim \{num_1', \ldots, num_m'\}} \quad \text{SUBTYPE\_BASE\_FINITE\_SET\_SUBSET}$$

$$\frac{}{\Theta \vdash \{num_1, \ldots, num_n\} \lesssim \mathbf{int}} \quad \text{SUBTYPE\_BASE\_FINITE\_SET\_INT}$$

$\boxed{\Theta; \Gamma \vdash \tau_1^p \lesssim \tau_2^p}$ Subtyping

$$\frac{\begin{array}{l}\Theta \vdash b_1^p \lesssim b_2^p \\ x = \mathbf{fresh}\,\Gamma \\ \Theta; \Gamma, x : b_1^p[\phi_1^p[x/z_1]] \models \phi_2^p[x/z_2]\end{array}}{\Theta; \Gamma \vdash \{z_1 : b_1^p | \phi_1^p\} \lesssim \{z_2 : b_2^p | \phi_2^p\}} \quad \text{SUBTYPE\_SUBTYPE}$$

$\boxed{\Theta; \Gamma \vdash_v v^p \Rightarrow \tau^p}$ Type synthesis for values

Infer that type of v is τ where x is a fresh variable representing v and γ a list of new bindings that will include the one for x

$$\frac{x : b^p[\phi^p] \in \Gamma}{\Theta; \Gamma \vdash_v x \Rightarrow \{z : b^p | \phi^p[z/x]\}} \quad \text{INFER\_V\_VAR}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v \mathbf{true} \Rightarrow \{z : \mathbf{bool} | z = \mathbf{true}\}} \quad \text{INFER\_V\_TRUE}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v \mathbf{false} \Rightarrow \{z : \mathbf{bool} | z = \mathbf{false}\}} \quad \text{INFER\_V\_FALSE}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v num \Rightarrow \{z : \mathbf{int} | z = num\}} \quad \text{INFER\_V\_NUM}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v \mathbf{bitone} \Rightarrow \{z : \mathbf{bit} | z = \mathbf{bitone}\}} \quad \text{INFER\_V\_BITONE}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v \mathbf{bitzero} \Rightarrow \{z : \mathbf{bit} | z = \mathbf{bitzero}\}} \quad \text{INFER\_V\_BITZERO}$$

$$\frac{x = \mathbf{fresh}\,\Gamma}{\Theta; \Gamma \vdash_v () \Rightarrow \{z : \mathbf{unit} | z = ()\}} \quad \text{INFER\_V\_UNIT}$$

$$\frac{\begin{array}{l}\Theta;\Gamma\vdash_v v_1^p \Rightarrow \{z_1 : b_1^p|\phi_1^p\} \quad .. \quad \Theta;\Gamma\vdash_v v_n^p \Rightarrow \{z_n : b_n^p|\phi_n^p\}\\ b^p = \textbf{single\_base}\, b_1^p\, ..\, b_n^p\\ \textbf{default}\, order \in \Theta\end{array}}{\Theta;\Gamma\vdash_v [v_1^p, .., v_n^p] \Rightarrow \{z : \textbf{vec}\, order\, b^p | z = [v_1^p, .., v_n^p]\}}\ \text{INFER\_V\_BITVEC}$$

$$\frac{\begin{array}{l}\Theta;\Gamma\vdash_v v_1^p \Rightarrow \tau_1^p \quad .. \quad \Theta;\Gamma\vdash_v v_n^p \Rightarrow \tau_n^p\\ b_1^p\, ..\, b_n^p = \textbf{b\_of}\,(\tau_1^p\, ..\, \tau_n^p)\end{array}}{\Theta;\Gamma\vdash_v (v_1^p, .., v_n^p) \Rightarrow \{z : (b_1^p, .., b_n^p) | z = (v_1^p, .., v_n^p)\}}\ \text{INFER\_V\_TUPLE}$$

$$\frac{\begin{array}{l}x = \textbf{fresh}\,\Gamma\\ \Theta;\Gamma\vdash_v v_1^p \Rightarrow \{z : b^p|\phi_1^p\}\\ \Theta;\Gamma\vdash_v v_2^p \Rightarrow \{z : \textbf{list}\, b^p|\phi_2^p\}\end{array}}{\Theta;\Gamma\vdash_v \textbf{cons}\, v_1^p\, v_2^p \Rightarrow \{z : \textbf{list}\, b^p | z = \textbf{cons}\, v_1^p\, v_2^p\}}\ \text{INFER\_V\_LIST\_CONS}$$

$$\frac{\begin{array}{l}\Theta(\dot{ctor} \mapsto b^p\{z : b_2^p|\phi_2^p\})\\ \Theta;\Gamma\vdash_v v^p \Rightarrow \{z : b_1^p|\phi_1^p\}\\ \sigma = \textbf{UNIFY}\, b_1^p\, b_2^p\\ \Theta;\Gamma\vdash \{z : b_1^p[\sigma]|\phi_1^p\} \lesssim \{z : b_2^p[\sigma]|\phi_2^p\}\end{array}}{\Theta;\Gamma\vdash_v \dot{ctor}\, v^p \Rightarrow \{z : b^p[\sigma] | z = \dot{ctor}\, v^p\}}\ \text{INFER\_V\_CONSTR}$$

$\boxed{\Theta;\Gamma\vdash_v v^p \Leftarrow \tau^p}$ Type check v^p is τ^p where Γ is an updated context.

$$\frac{\begin{array}{l}\Theta;\Gamma\vdash_v v^p \Rightarrow \tau_1^p\\ \Theta;\Gamma,\gamma^p\vdash \tau_1^p \lesssim \tau_2^p\end{array}}{\Theta;\Gamma\vdash_v v^p \Leftarrow \tau_2^p}\ \text{CHECK\_V\_V}$$

$\boxed{\Theta;\Gamma\vdash pat^p \Rightarrow \tau^p \leadsto x;\gamma^p; xlist}$ Infer type of patmtern pat^p. $xlist$ is the list of pattern variables in the patte

$$\frac{\begin{array}{l}x = \textbf{fresh}\,\Gamma\\ \Theta;\Gamma, x : b^p[\phi^p[x/z]]\vdash pat^p \Leftarrow \{z : b^p|\phi^p\} \leadsto x;\gamma^p; x_1, .., x_n\end{array}}{\Theta;\Gamma\vdash (\{z : b^p|\phi^p\})pat^p \Rightarrow \{z : b^p|\phi^p\} \leadsto x; x : b^p[\phi^p[x/z]], \gamma^p; x_1, .., x_n}\ \text{INFER\_PATM\_TYP}$$

$\boxed{\Theta;\Gamma\vdash pat_1^p\, ..\, pat_n^p \Leftarrow b_1^p, .., b_m^p \leadsto \gamma^p; x_1\, ..\, x_j}$ Type check of list of patterns

$$\frac{}{\Theta;\Gamma\vdash \Leftarrow \leadsto ;}\ \text{CHECK\_PATMS\_NIL}$$

$$\frac{\begin{array}{l}\Theta;\Gamma\vdash pat^p \Leftarrow \{z : b^p|\top\} \leadsto x;\gamma^p; xlist\\ \Theta;\Gamma,\gamma^p\vdash pat_1^p\, ..\, pat_n^p \Leftarrow b_1^p, .., b_n^p \leadsto \gamma_2^p; x_1\, ..\, x_n\end{array}}{\Theta;\Gamma\vdash pat^p\, pat_1^p\, ..\, pat_n^p \Leftarrow b^p, b_1^p, .., b_n^p \leadsto (\gamma^p, \gamma_2^p); x\, x_1\, ..\, x_n}\ \text{CHECK\_PATMS\_CONS}$$

$\boxed{\Theta;\Gamma\vdash pat^p \Leftarrow \tau^p \leadsto x;\gamma^p; xlist}$ Type check pattern pat^p is τ^p. $xlist$ is the list of pattern variables in the pat

$$\frac{}{\Theta;\Gamma\vdash \_ \Leftarrow \{z : b^p|\phi^p\} \leadsto x; ;}\ \text{CHECK\_PATM\_WILD}$$

$$\frac{}{\Theta;\Gamma\vdash id \Leftarrow \{z : b^p|\phi^p\} \leadsto x; id : b^p[x = id]; id}\ \text{CHECK\_PATM\_ID}$$

$$\frac{x_2 = \textbf{fresh}\,\Gamma}{\Theta;\Gamma\vdash lit \Leftarrow \{z : b^p|\phi^p\} \leadsto x_1; x_2 : b^p[x_1 = lit]; x_1}\ \text{CHECK\_PATM\_LIT}$$

$$\frac{\begin{array}{l}\Theta(id \mapsto \tau_1^p, \{z : b^p|\phi^p\})\\ x_2 = \textbf{fresh}\,\Gamma\\ \Theta;\Gamma, x_2 : b^p[\phi^p[x_2/z]]\vdash (pat_1^p, .., pat_n^p) \Leftarrow \{z : b^p|\phi^p\} \leadsto x_2;\gamma^p; xlist\\ v^p = \textbf{mk\_ctor\_v}\, id\, xlist\\ \Theta;\Gamma\vdash \tau_2^p \lesssim \tau_1^p\end{array}}{\Theta;\Gamma\vdash id(pat_1^p, .., pat_n^p) \Leftarrow \tau_2^p \leadsto x_2; x_2 : b^p[\phi^p[x_2/z] \wedge x_2 = v^p], \gamma^p; xlist}\ \text{CHECK\_PATM\_CTOR}$$

$$\frac{\begin{array}{l}\Theta(id \mapsto \tau_1^p, \{z : b^p | \phi^p\}) \\ x_2 = \textbf{fresh}\,\Gamma \\ \Theta; \Gamma, x_2 : b^p[\phi^p[x_2/z]] \vdash pat^p \Leftarrow \{z : b^p | \phi^p\} \leadsto x_2; \gamma^p; xlist \\ v^p = \textbf{mk\_ctor\_v}\; id\; xlist \\ \Theta; \Gamma \vdash \tau_2^p \lesssim \tau_1^p\end{array}}{\Theta; \Gamma \vdash id(pat^p) \Leftarrow \tau_2^p \leadsto x_2; x_2 : b^p[\phi^p[x_2/z] \wedge x_2 = v^p], \gamma^p; xlist} \quad \text{CHECK\_PATM\_CTOR\_SINGLE}$$

$$\frac{\begin{array}{c}x_1 \mathrel{..} x_n, \gamma_1^p = \textbf{mk\_proj\_vars}\; x\; b_1^p, \mathrel{..}, b_n^p \\ \Theta; \Gamma, \gamma_1^p \vdash pat_1^p \mathrel{..} pat_n^p \Leftarrow b_1^p, \mathrel{..}, b_n^p \leadsto \gamma_2^p; x_1 \mathrel{..} x_n\end{array}}{\Theta; \Gamma \vdash (pat_1^p, \mathrel{..}, pat_n^p) \Leftarrow \{z : (b_1^p, \mathrel{..}, b_n^p) | \phi^p\} \leadsto x; (\gamma_1^p, \gamma_2^p); x} \quad \text{CHECK\_PATM\_TUPLE}$$

$$\frac{\begin{array}{l}\Theta; \Gamma \vdash pat^p \Leftarrow \tau_1^p \leadsto x; \gamma^p; xlist \\ \Theta; \Gamma, \gamma^p \vdash \tau_1^p \lesssim \tau_2^p\end{array}}{\Theta; \Gamma \vdash (\tau_1^p) pat^p \Leftarrow \tau_2^p \leadsto x; \gamma^p; xlist} \quad \text{CHECK\_PATM\_TYP}$$

$\boxed{\Theta; \Gamma; \Delta \vdash lexp^p \Rightarrow \tau^p \leadsto \Delta'}$ Type synthesis for l-value expression $lexp^p$

$$\frac{u : \tau^p \in \Delta}{\Theta; \Gamma; \Delta \vdash u \Rightarrow \tau^p \leadsto \Delta} \quad \text{INFER\_LEXP\_VAR\_BOUND}$$

$$\frac{u \notin \Delta}{\Theta; \Gamma; \Delta \vdash (\tau^p) u \Rightarrow \tau^p \leadsto \Delta + u : \tau^p} \quad \text{INFER\_LEXP\_CAST\_NOT\_BOUND}$$

$$\frac{\begin{array}{l}u : \tau_1^p \in \Delta \\ \Theta; \Gamma \vdash \tau_2^p \lesssim \tau_1^p\end{array}}{\Theta; \Gamma; \Delta \vdash (\tau_2^p) u \Rightarrow \tau_2^p \leadsto \Delta + +u : \tau_2^p} \quad \text{INFER\_LEXP\_CAST\_BOUND}$$

$$\frac{\Theta; \Gamma; \Delta \vdash (lexp_1^p \mathrel{..} lexp_n^p) \Rightarrow (\{z_1 : b_1^p | \phi_1^p\} \mathrel{..} \{z_n : b_n^p | \phi_n^p\}) \leadsto \Delta'}{\Theta; \Gamma; \Delta \vdash (lexp_1^p, \mathrel{..}, lexp_n^p) \Rightarrow \{z : (b_1^p, \mathrel{..}, b_n^p) | \phi_1^p[(\textbf{proj}\; p_1\; z)/z_1] \wedge \mathrel{..} \wedge \phi_n^p[(\textbf{proj}\; p_n\; z)/z_n]\} \leadsto \Delta'} \quad \text{INFER\_LEXP\_T}$$

$$\frac{\begin{array}{l}b^p\; \tau_1^p = \textbf{lookup\_field\_record\_type}\; \Theta\; id \\ \Theta; \Gamma; \Delta \vdash lexp^p \Rightarrow \tau_2^p \leadsto \Delta' \\ \Theta; \Gamma \vdash \tau_2^p \lesssim \tau_1^p\end{array}}{\Theta; \Gamma; \Delta \vdash lexp^p.id \Rightarrow \{z : b^p | \top\} \leadsto \Delta'} \quad \text{INFER\_LEXP\_FIELD}$$

$\boxed{\Theta; \Gamma; \Delta \vdash (lexp_1^p \mathrel{..} lexp_n^p) \Rightarrow (\tau_1^p \mathrel{..} \tau_m^p) \leadsto \Delta'}$ Type synthesis for a list of l-value expressions with threading of

$$\frac{}{\Theta; \Gamma; \Delta \vdash (\,) \Rightarrow (\,) \leadsto \Delta} \quad \text{INFER\_LEXPS\_NIL}$$

$$\frac{\begin{array}{l}\Theta; \Gamma; \Delta \vdash lexp^p \Rightarrow \tau^p \leadsto \Delta' \\ \Theta; \Gamma; \Delta' \vdash (lexp_1^p \mathrel{..} lexp_n^p) \Rightarrow (\tau_1^p \mathrel{..} \tau_n^p) \leadsto \Delta''\end{array}}{\Theta; \Gamma; \Delta \vdash (lexp^p\; lexp_1^p \mathrel{..} lexp_n^p) \Rightarrow (\tau^p\; \tau_1^p \mathrel{..} \tau_n^p) \leadsto \Delta''} \quad \text{INFER\_LEXPS\_CONS}$$

$\boxed{\Theta; \Gamma; \Delta \vdash (lexp_1^p \mathrel{..} lexp_n^p) \Leftarrow (b_1^p \mathrel{..} b_m^p) \leadsto \Delta'}$ Type check for a list l-value expressions with threading of the Γ

$$\frac{}{\Theta; \Gamma; \Delta \vdash (\,) \Leftarrow (\,) \leadsto \Delta} \quad \text{CHECK\_LEXPS\_NIL}$$

$$\frac{\begin{array}{l}\Theta; \Gamma; \Delta \vdash lexp^p \Leftarrow \{z : b^p | \top\} \leadsto \Delta' \\ \Theta; \Gamma; \Delta' \vdash (lexp_1^p \mathrel{..} lexp_n^p) \Leftarrow (b_1^p \mathrel{..} b_n^p) \leadsto \Delta''\end{array}}{\Theta; \Gamma; \Delta \vdash (lexp^p\; lexp_1^p \mathrel{..} lexp_n^p) \Leftarrow (b^p\; b_1^p \mathrel{..} b_n^p) \leadsto \Delta''} \quad \text{CHECK\_LEXPS\_CONS}$$

$\boxed{\Theta; \Gamma; \Delta \vdash lexp^p \Leftarrow \tau^p \leadsto \Delta'}$ Type check l-value expression $lexp^p$

$$\frac{u \notin \Delta}{\Theta; \Gamma; \Delta \vdash u \Leftarrow \tau^p \leadsto \Delta + u : \tau^p} \quad \text{CHECK\_LEXP\_VAR\_NOT\_BOUND}$$

$$\frac{\begin{array}{l}\Theta;\Gamma;\Delta\vdash lexp^p\Rightarrow\tau_2^p\leadsto\Delta'\\ \Theta;\Gamma\vdash\tau_1^p\lesssim\tau_2^p\end{array}}{\Theta;\Gamma;\Delta\vdash lexp^p\Leftarrow\tau_1^p\leadsto\Delta'}\quad\text{CHECK\_LEXP\_SUBTYPE\_INFER}$$

$$\frac{\Theta;\Gamma;\Delta\vdash(lexp_1^p\,..\,lexp_n^p)\Leftarrow(b_1^p\,..\,b_n^p)\leadsto\Delta'}{\Theta;\Gamma;\Delta\vdash(lexp_1^p,\,..\,,lexp_n^p)\Leftarrow\{z:(b_1^p,\,..\,,b_n^p)|\phi^p\}\leadsto\Delta'}\quad\text{CHECK\_LEXP\_TUPLE}$$

$\boxed{\Theta;\Phi;\Gamma;\Delta\vdash pexp^p\Leftarrow\tau_1^p,\tau_2^p\leadsto\Gamma'}$ Type check pattern is τ_1^p and the expression is τ_2^p.

$$\frac{\begin{array}{l}x=\textbf{fresh}\,\Gamma\\ \Theta;\Gamma,x:b^p[\phi^p[x/z]]\vdash pat^p\Leftarrow\{z:b^p|\phi^p\}\leadsto x;\gamma^p;x_1,\,..\,,x_n\\ \Theta;\Phi;\Gamma,x:b^p[\phi^p[x/z]],\gamma^p;\Delta\vdash e^p\Leftarrow\tau^p\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash pat^p\Rightarrow e^p\Leftarrow\{z:b^p|\phi^p\},\tau^p\leadsto\Gamma,x:b^p[\phi^p[x/z]],\gamma^p}\quad\text{CHECK\_PEXP\_EXP}$$

$$\frac{\begin{array}{l}x=\textbf{fresh}\,\Gamma\\ \Theta;\Gamma\vdash pat^p\Leftarrow\tau_2^p\leadsto x;\gamma^p;x_1,\,..\,,x_n\\ \Theta;\Phi;\Gamma,\gamma^p\vdash_e e_1^p\Rightarrow\{z:\textbf{bool}|\phi^p\}\leadsto x';\gamma_2^p\\ \Theta;\Phi;(\Gamma,\gamma^p),\gamma_2^p;\Delta\vdash e_2^p\Leftarrow\tau_1^p\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash pat^p\ \textbf{when}\ e_1^p\Rightarrow e_2^p\Leftarrow\tau_2^p,\tau_1^p\leadsto(\Gamma,\gamma^p),\gamma_2^p}\quad\text{CHECK\_PEXP\_WHEN}$$

$\boxed{\textbf{match\_overload}\,\Theta\,\Gamma\{z_2:b_2^p|\phi_2^p\}(a_1^p,\,...\,,a_n^p)(a^p)\sigma}$ Find list of functions having input types that unify with p

$$\frac{\begin{array}{l}\sigma=\textbf{UNIFY}\,b_1^p\,b_2^p\\ \Theta;\Gamma\vdash\{z_2:b_2^p[\sigma]|\phi_2^p\}\lesssim\{x_1:b_1^p[\sigma]|\phi_1^p\}\end{array}}{\textbf{match\_overload}\,\Theta\,\Gamma\{z_2:b_2^p|\phi_2^p\}(x_1:b_1^p[\phi_1^p]->\{z_3:b_3^p|\phi_3^p\},a_1^p,\,..\,,a_n^p)(x_1:b_1^p[\phi_1^p]->\{z_3:b_3^p|\phi_3^p\})\sigma}\quad\text{MATCH\_A}$$

$$\frac{\textbf{match\_overload}\,\Theta\,\Gamma\{z_2:b_2^p|\phi_2^p\}(a_1^p,\,..\,,a_n^p)(x_1:b_1^p[\phi_1^p]->\{z_3:b_3^p|\phi_3^p\})\sigma}{\textbf{match\_overload}\,\Theta\,\Gamma\{z_2:b_2^p|\phi_2^p\}(a^p,a_1^p,\,..\,,a_n^p)(x_1:b_1^p[\phi_1^p]->\{z_3:b_3^p|\phi_3^p\})\sigma}\quad\text{MATCH\_ARG\_TAIL}$$

$\boxed{\Theta;\Phi;\Gamma;\Delta\vdash(a_1^p,\,...\,,a_n^p)e^p\Rightarrow\tau^p\leadsto x;\gamma^p}$ Type synthesis for overloaded function application

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\{z_2:b_2^p|\phi_2^p\}\leadsto x_2;\gamma^p\\ \textbf{match\_overload}\,\Theta\,\Gamma,\gamma^p\{z_2:b_2^p|\phi_2^p\}(a_1^p,\,...\,,a_n^p)(x_1:b_1^p[\phi_1^p]->\{z_3:b_3^p|\phi_3^p\})\sigma\\ x_3=\textbf{fresh}\,\Gamma,\gamma^p\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash(a_1^p,\,...\,,a_n^p)e^p\Rightarrow\{z_3:b_3^p[\sigma]|\phi_3^p[x_1/x_2]\}\leadsto x_3;x_3:b_3^p[\sigma][(\phi_3^p[x_3/z_3])[x_1/x_2]],\gamma^p}\quad\text{INFER\_APP\_APP\_HEA}$$

$\boxed{\Theta;\Phi;\Gamma;\Delta\vdash e_1^p\,..\,e_m^p\Rightarrow\tau_1^p\,..\,\tau_n^p\leadsto x_1\,..\,x_j;\gamma^p}$ Type synthesis for expressions types.

$$\frac{}{\Theta;\Phi;\Gamma;\Delta\vdash\ \Rightarrow\ \leadsto\ ;\epsilon}\quad\text{INFER\_E\_LIST\_NIL}$$

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\tau^p\leadsto x;\gamma_1^p\\ \Theta;\Phi;\Gamma,\gamma_1^p;\Delta\vdash e_1^p\,..\,e_n^p\Rightarrow\tau_1^p\,..\,\tau_n^p\leadsto x_1\,..\,x_n;\gamma_2^p\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash e^p\,e_1^p\,..\,e_n^p\Rightarrow\tau^p\,\tau_1^p\,..\,\tau_n^p\leadsto x\,x_1\,..\,x_n;(\gamma_1^p,\gamma_2^p)}\quad\text{INFER\_E\_LIST\_CONS}$$

$\boxed{\Theta;\Phi;\Gamma;\Delta\vdash letbind\leadsto\gamma^p}$ Bindings γ for a let-bind

Either infer type of expression and then check pattern has the type or if pattern is type-pattern, check pattern and check expression.

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\tau^p\leadsto x_1;\gamma_1^p\\ \Theta;\Gamma,\gamma_1^p\vdash pat^p\Leftarrow\tau^p\leadsto x_1;\gamma_2^p;y_1,\,..\,,y_n\\ y_1\,..\,y_n\notin\Gamma\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash\textbf{let}\,pat^p=e^p\leadsto(\gamma_1^p,\gamma_2^p)}\quad\text{LETBIND\_INFER}$$

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma;\epsilon\vdash e^p \Leftarrow \{z:b^p|\phi^p\}\\ x=\ \mathbf{fresh}\,\Gamma\\ \Theta;\Gamma,x:b^p[\phi^p[x/z]]\vdash pat^p \Leftarrow \{z:b^p|\phi^p\}\leadsto x;\gamma^p;x_1,..,x_n\\ x_1\,..\,x_n\ \notin\ \Gamma\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash \mathbf{let}\,(\{z:b^p|\phi^p\})pat^p = e^p \leadsto \gamma^p}\ \textsc{letbind\_check}$$

$\boxed{\Theta;\Phi;\Gamma\vdash_e e^p \Rightarrow \tau^p \leadsto x;\gamma^p}$ Infer that type of e^p is τ^p. γ^p are new fresh variables to capture types of subterr

$$\frac{\begin{array}{l}x=\ \mathbf{fresh}\,\Gamma\\ \Theta;\Gamma\vdash_v v^p\Rightarrow\{z:b^p|\phi^p\}\end{array}}{\Theta;\Phi;\Gamma\vdash_e v^p\Rightarrow\{z:b^p|\phi^p\}\leadsto x;x:b^p[\phi^p[x/z]]}\ \textsc{infer\_e\_val}$$

$$\frac{\begin{array}{l}x=\ \mathbf{fresh}\,\Gamma\\ u:\{z:b^p|\phi^p\}\in\Delta\end{array}}{\Theta;\Phi;\Gamma\vdash_e u\Rightarrow\{z:b^p|\phi^p\}\leadsto x;x:b^p[\phi^p[x/z]]}\ \textsc{infer\_e\_mvar}$$

$$\frac{x=\ \mathbf{fresh}\,\Gamma}{\Theta;\Phi;\Gamma\vdash_e \mathbf{sizeof}\,ce^p\Rightarrow\{z:\mathbf{int}|z=ce^p\}\leadsto x;x:\mathbf{int}[x=ce^p]}\ \textsc{infer\_e\_sizeof}$$

$$\frac{x=\ \mathbf{fresh}\,\Gamma}{\Theta;\Phi;\Gamma\vdash_e \mathbf{constraint}\,\phi^p\Rightarrow\{z:\mathbf{bool}|\phi^p\}\leadsto x;x:\mathbf{bool}[\phi^p]}\ \textsc{infer\_e\_constraint}$$

$$\frac{\begin{array}{l}a_1^p,\,...\,,a_n^p=\ \mathbf{lookup\_fun\_type}\,\Theta\,\Phi\,fp\\ \Theta;\Phi;\Gamma;\Delta\vdash(a_1^p,\,...\,,a_n^p)e^p\Rightarrow\tau_1^p\leadsto x_1;\gamma^p\end{array}}{\Theta;\Phi;\Gamma\vdash_e fp\,e^p\Rightarrow\tau_1^p\leadsto x_1;\gamma^p}\ \textsc{infer\_e\_app}$$

$$\frac{\begin{array}{l}x=\ \mathbf{fresh}\,\Gamma,\gamma^p\\ \Theta;\Phi;\Gamma;\Delta\vdash e_1^p\,..\,e_n^p\Rightarrow\tau_1^p\,..\,\tau_n^p\leadsto x_1\,..\,x_n;\gamma^p\\ b_1^p\,..\,b_n^p=\ \mathbf{b\_of}\,(\tau_1^p\,..\,\tau_n^p)\end{array}}{\Theta;\Phi;\Gamma\vdash_e(e_1^p,\,..\,,e_n^p)\Rightarrow\{z:(b_1^p,\,..\,,b_n^p)|z=(x_1,\,..\,,x_n)\}\leadsto x;x:(b_1^p,\,..\,,b_n^p)[x=(x_1,\,..\,,x_n)],\gamma^p}\ \textsc{infer\_e\_tupl}$$

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma\vdash_e e_1^p\Rightarrow\{z:\mathbf{vec}\,order\,b^p|\phi_1^p\}\leadsto x_1;\gamma_1^p\quad ...\quad \Theta;\Phi;\Gamma\vdash_e e_n^p\Rightarrow\{z:\mathbf{vec}\,order\,b^p|\phi_n^p\}\leadsto x_n;\gamma_n^p\\ x=\ \mathbf{fresh}\,\Gamma,(\gamma_1^p\,...\,\gamma_n^p)\end{array}}{\Theta;\Phi;\Gamma\vdash_e[e_1^p;\,...\,;e_n^p]\Rightarrow\{z:\mathbf{vec}\,order\,b^p|\mathbf{len}\,z=\mathbf{sum}\,(x_1\,...\,x_n)\}\leadsto x;x:\mathbf{vec}\,order\,b^p[\mathbf{len}\,x=\mathbf{sum}\,(x_1\,...\,x_n)],(}$$

$$\frac{\begin{array}{l}b^p\,\tau_1^p=\ \mathbf{lookup\_field\_record\_type}\,\Theta\,field\\ \Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\tau_2^p\leadsto x_1;\gamma^p\\ \Theta;\Gamma,\gamma^p\vdash\tau_2^p\lesssim\tau_1^p\\ x_2=\ \mathbf{fresh}\,\Gamma,\gamma^p\end{array}}{\Theta;\Phi;\Gamma\vdash_e e^p.field\Rightarrow\{z:b^p|x_1.field=z\}\leadsto x_2;x_2:b^p[x_1.field=x_2],\gamma^p}\ \textsc{infer\_e\_field\_access}$$

$$\frac{\begin{array}{l}x=\ \mathbf{fresh}\,\Gamma,\gamma_1^p\\ \Theta;\Phi;\Gamma\vdash_e e_1^p\Rightarrow\{z:\mathbf{bool}|\phi^p\}\leadsto x_1;\gamma_1^p\\ \Theta;\Phi;\Gamma,\gamma_1^p;\Delta\vdash e_2^p\Leftarrow\{z:\mathbf{unit}|\top\}\end{array}}{\Theta;\Phi;\Gamma\vdash_e loop\,e_1^p\,e_2^p\Rightarrow\{z:\mathbf{unit}|\top\}\leadsto x;x:\mathbf{unit}[\top]}\ \textsc{infer\_e\_loop}$$

$$\frac{\begin{array}{l}\Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\{z:b^p|\phi^p\}\leadsto x;\gamma^p\\ b_1^p\,..\,b_n^p=\ \mathbf{lookup\_types\_for}\,b^p\,field_1\,..\,field_n\\ \Theta;\Phi;\Gamma,\gamma^p;\Delta\vdash e_1^p\,..\,e_n^p\Leftarrow\{z:b_1^p|\top\}\,..\,\{z:b_n^p|\top\}\leadsto\Gamma'\end{array}}{\Theta;\Phi;\Gamma\vdash_e\{e^p\,\mathbf{with}\,field_1=e_1^p;\,..\,;field_n=e_n^p\}\Rightarrow\{z:b^p|\phi^p\}\leadsto x;x:b^p[\phi^p[x/z]],\gamma^p}\ \textsc{infer\_e\_record\_update}$$

$\boxed{\Theta;\Phi;\Gamma;\Delta\vdash lexp^p=e^p\leadsto\Delta';\gamma^p}$ Assignment expression type checking.

$$\frac{\begin{array}{l}u\notin\Delta\\ \Theta;\Phi;\Gamma\vdash_e e^p\Rightarrow\tau^p\leadsto x;\gamma^p\end{array}}{\Theta;\Phi;\Gamma;\Delta\vdash u=e^p\leadsto\Delta+u:\tau^p;\gamma^p}\ \textsc{typing\_lexp\_mvar\_not\_bound}$$

$$\frac{\begin{array}{l} u : \tau^p \in \Delta \\ \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash u = e^p \leadsto \Delta;} \quad \text{TYPING\_LEXP\_MVAR\_BOUND}$$

$$\frac{\begin{array}{l} u \notin \Delta \\ \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash (\tau^p) u = e^p \leadsto \Delta + u : \tau^p;} \quad \text{TYPING\_LEXP\_CAST\_NOT\_BOUND}$$

$$\frac{\begin{array}{l} u : \tau_1^p \in \Delta \\ \Theta; \Gamma \vdash \tau_2^p \lesssim \tau_1^p \\ \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau_2^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash (\tau_2^p) u = e^p \leadsto \Delta + +u : \tau_2^p;} \quad \text{TYPING\_LEXP\_CAST\_BOUND}$$

$$\frac{\begin{array}{l} \tau_1^p\, \tau_2^p = \mathbf{lookup\_field\_and\_record\_type}\, \Theta\, u \\ u : \tau^p \in \Delta \\ \Theta; \Gamma \vdash \tau^p \lesssim \tau_2^p \\ \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau_1^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash u.id = e^p \leadsto \Delta;} \quad \text{TYPING\_LEXP\_FIELD}$$

$$\frac{\Theta; \Phi; \Gamma; \Delta \vdash lexp^p = e^p \leadsto \Delta'; \gamma^p}{\Theta; \Phi; \Gamma; \Delta \vdash (lexp^p) = (e^p) \leadsto \Delta; \gamma^p} \quad \text{TYPING\_LEXP\_TUPLE\_SINGLE}$$

$$\frac{\begin{array}{l} \Theta; \Phi; \Gamma; \Delta \vdash lexp^p = e^p \leadsto \Delta'; \gamma_1^p \\ \Theta; \Phi; \Gamma; \Delta' \vdash (lexp_1^p, \,..\,, lexp_n^p) = (e_1^p, \,..\,, e_n^p) \leadsto \Delta''; \gamma_2^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash (lexp^p, lexp_1^p, \,..\,, lexp_n^p) = (e^p, e_1^p, \,..\,, e_n^p) \leadsto \Delta''; (\gamma_1^p, \gamma_2^p)} \quad \text{TYPING\_LEXP\_TUPLE\_CONS}$$

$\boxed{\Theta; \Phi; \Gamma; \Delta \vdash e_1^p \,..\, e_n^p \Leftarrow \tau_1^p \,..\, \tau_m^p \leadsto \Gamma'}$ Type check list of expressions with context threaded through

$$\frac{}{\Theta; \Phi; \Gamma; \Delta \vdash \Leftarrow \leadsto \Gamma} \quad \text{CHECK\_E\_LIST\_NIL}$$

$$\frac{\begin{array}{l} \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau^p \\ \Theta; \Phi; \Gamma; \Delta \vdash e_1^p \,..\, e_n^p \Leftarrow \tau_1^p \,..\, \tau_n^p \leadsto \Gamma'' \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash e^p\, e_1^p \,..\, e_n^p \Leftarrow \tau^p\, \tau_1^p \,..\, \tau_n^p \leadsto \Gamma''} \quad \text{CHECK\_E\_LIST\_CONS}$$

$\boxed{\Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau^p}$ Type check of e^p against τ^p

$$\frac{\begin{array}{l} \tau_1^p = \mathbf{lookup\_return}\, \Gamma \\ \Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \tau_1^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash \mathbf{return}\, e^p \Leftarrow \tau_2^p} \quad \text{CHECK\_E\_RETURN}$$

$$\frac{\Theta; \Phi; \Gamma; \Delta \vdash e^p \Leftarrow \{z : \mathbf{unit} | \top\}}{\Theta; \Phi; \Gamma; \Delta \vdash \mathbf{exit}\, e^p \Leftarrow \tau^p} \quad \text{CHECK\_E\_EXIT}$$

$$\frac{\begin{array}{l} \Theta; \Phi; \Gamma \vdash_e e^p \Rightarrow \{z_3 : \mathbf{bool} | \phi_1^p\} \leadsto x; \gamma^p \\ x_3 = \mathbf{fresh}\, \Gamma, \gamma^p \\ \Theta; \Phi; \Gamma, (\gamma^p); \Delta \vdash e_1^p \Leftarrow \{z : b^p | x = \mathbf{true} \Longrightarrow \phi^p\} \\ \Theta; \Phi; \Gamma, (\gamma^p); \Delta \vdash e_2^p \Leftarrow \{z : b^p | x = \mathbf{false} \Longrightarrow \phi^p\} \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash \mathbf{if}\, e^p\, \mathbf{then}\, e_1^p\, \mathbf{else}\, e_2^p \Leftarrow \{z : b^p | \phi^p\}} \quad \text{CHECK\_E\_IF}$$

$$\frac{\begin{array}{l} \Theta; \Phi; \Gamma; \Delta \vdash \mathbf{let}\, pat^p = e_1^p \leadsto \gamma^p \\ \Theta; \Phi; \Gamma, \gamma^p; \Delta \vdash e_2^p \Leftarrow \tau^p \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash \mathbf{let}\, pat^p = e_1^p\, \mathbf{in}\, e_2^p \Leftarrow \tau^p} \quad \text{CHECK\_E\_LET}$$

$$\frac{\begin{array}{l} \Theta; \Phi; \Gamma \vdash_e e^p \Rightarrow \tau^p \leadsto x; \gamma_1^p \\ \Theta; \Phi; \Gamma, \gamma_1^p; \Delta \vdash pat_1^p \Rightarrow e_1^p \Leftarrow \tau^p, \{z : b^p | \phi^p\} \leadsto \Gamma_1 \quad \ldots \quad \Theta; \Phi; \Gamma, \gamma_1^p; \Delta \vdash pat_n^p \Rightarrow e_n^p \Leftarrow \tau^p, \{z : b^p | \phi^p\} \leadsto \Gamma_n \end{array}}{\Theta; \Phi; \Gamma; \Delta \vdash \mathbf{match}\, e^p \{pat_1^p \Rightarrow e_1^p, \,\ldots, pat_n^p \Rightarrow e_n^p\} \Leftarrow \{z : b^p | \phi^p\}} \quad \text{CH}$$

$$\frac{\Theta;\Phi;\Gamma;\Delta \vdash lexp^p = e_1^p \leadsto \Delta';\gamma^p \quad \Theta;\Phi;\Gamma,\gamma^p;\Delta' \vdash e_2^p \Leftarrow \{z : b^p|\phi^p\}}{\Theta;\Phi;\Gamma;\Delta \vdash lexp^p := e_1^p\,\mathbf{in}\,e_2^p \Leftarrow \{z : b^p|\phi^p\}}\ \text{CHECK\_E\_ASSIGN}$$

$$\frac{\Theta;\Phi;\Gamma;\Delta \vdash e^p \Leftarrow \tau^p}{\Theta;\Phi;\Gamma;\Delta \vdash \{e^p\} \Leftarrow \tau^p}\ \text{CHECK\_E\_SEQ\_SINGLE}$$

$$\frac{\Theta;\Phi;\Gamma;\Delta \vdash e^p \Leftarrow \{z : \mathbf{unit}|\top\} \quad \Theta;\Phi;\Gamma;\Delta \vdash \{e_1^p;\,..\,;e_n^p\} \Leftarrow \tau^p}{\Theta;\Phi;\Gamma;\Delta \vdash \{e^p;e_1^p;\,..\,;e_n^p\} \Leftarrow \tau^p}\ \text{CHECK\_E\_SEQ\_CONS}$$

$$\frac{\begin{array}{l}\{z : \{field_1' : b_1^p,\,..\,,field_n' : b_n^p\}|\phi_2^p\} = \mathbf{lookup\_fields}\,\Theta\,field_1\,..\,field_n\\ \Theta;\Phi;\Gamma;\Delta \vdash e_1^p\,..\,e_n^p \Leftarrow \{z : b_1^p|\top\}\,..\,\{z : b_n^p|\top\} \leadsto \Gamma'\\ \Theta;\Gamma \vdash \{z : b^p|\phi^p\} \lesssim \{z_2 : \{field_1 : b_1^p,\,..\,,field_n : b_n^p\}|\phi_2^p\}\end{array}}{\Theta;\Phi;\Gamma;\Delta \vdash \{field_1 = e_1^p,\,..\,,field_n = e_n^p\} \Leftarrow \{z : b^p|\phi^p\}}\ \text{CHECK\_E\_RECORD}$$

$$\frac{\Theta;\Phi;\Gamma \vdash_e e^p \Rightarrow \tau_1^p \leadsto x;\gamma^p \quad \Theta;\Gamma,\gamma^p \vdash \tau_1^p \lesssim \tau_2^p}{\Theta;\Phi;\Gamma;\Delta \vdash e^p \Leftarrow \tau_2^p}\ \text{CHECK\_E\_SUBTYPE}$$

$$\boxed{\Theta;\Phi;\Gamma \vdash funcl_1\,..\,funcl_n \Leftarrow x : b^p[\phi^p],\tau_2^p \leadsto \Phi';\Gamma'}$$

$$\frac{}{\Theta;\Phi;\Gamma \vdash \Leftarrow x : b^p[\phi^p],\tau_2^p \leadsto \Phi;\Gamma}\ \text{CHECK\_FUNCLSNIL}$$

$$\frac{\begin{array}{l}x_2 = \mathbf{fresh}\,\Gamma\\ \Gamma' = \mathbf{add\_return}\,\Gamma\,\tau_2^p\\ \Theta;\Phi;\Gamma';\epsilon \vdash pexp^p[x_2/x] \Leftarrow \{z : b^p|\phi^p[z/x]\},\tau_2^p[x_2/x] \leadsto \Gamma''\\ \Phi'' = \mathbf{add\_fun}\,\Phi\,x\,b^p\,\phi^p\,\tau_2^p\,id\,pexp^p\\ \Theta;\Phi'';\Gamma \vdash funcl_1\,...\,funcl_n \Leftarrow x : b^p[\phi^p],\tau_2^p \leadsto \Phi''';\Gamma'''\end{array}}{\Theta;\Phi;\Gamma \vdash id\,pexp^p\,funcl_1\,...\,funcl_n \Leftarrow x : b^p[\phi^p],\tau_2^p \leadsto \Phi''';\Gamma}\ \text{CHECK\_FUNCLSCONS}$$

$$\boxed{\Theta;\Phi;\Gamma \vdash \mathrm{def}^p \leadsto \Phi';\Gamma'}$$

$$\frac{\Theta;\Phi;\Gamma \vdash funcl_1\,..\,funcl_n \Leftarrow x : b^p[\phi^p],\tau_2^p \leadsto \Phi';\Gamma'}{\Theta;\Phi;\Gamma \vdash \mathbf{function}\,x : b^p[\phi^p]-> \tau_2^p\,funcl_1\,\mathbf{and}\,..\,\mathbf{and}\,funcl_n \leadsto \Phi';\Gamma'}\ \text{CHECK\_DEF\_FUNDEF}$$

$$\frac{\Theta;\Phi;\Gamma;\epsilon \vdash \mathbf{let}\,pat^p = e^p \leadsto \gamma^p}{\Theta;\Phi;\Gamma \vdash \mathbf{let}\,pat^p = e^p \leadsto \Phi;\Gamma,\gamma^p}\ \text{CHECK\_DEF\_LET}$$

$$\frac{id : a^p \notin \Phi}{\Theta;\Phi;\Gamma \vdash \mathbf{val}\,id : a^p \leadsto \Phi, id : a^p;\Gamma}\ \text{CHECK\_DEF\_VAL}$$

$$\frac{}{\Theta;\Phi;\Gamma \vdash \mathbf{overload}\,id[id_1;\,...\,;id_n] \leadsto \Phi, id[id_1\,...\,id_n];\Gamma}\ \text{CHECK\_DEF\_OVERLOAD}$$

$$\frac{}{\Theta;\Phi;\Gamma \vdash \mathbf{default}\,order \leadsto \Phi;\Gamma}\ \text{CHECK\_DEF\_DEFAULT}$$

$$\frac{}{\Theta;\Phi;\Gamma \vdash \mathbf{typedef}\,id = \forall\,\overline{kp_i : b_i^p[\phi_i^p]}^{\,i}\,\tau^p \leadsto \Phi;\Gamma}\ \text{CHECK\_DEF\_TYPEDEF}$$

$$\boxed{\Theta;\Phi;\Gamma \vdash \mathrm{def}_1^p\,...\,\mathrm{def}_n^p \leadsto \Theta';\Phi';\Gamma'}$$

$$\frac{\Theta;\Phi;\Gamma \vdash \mathrm{def}^p \leadsto \Phi';\Gamma'}{\Theta;\Phi;\Gamma \vdash \mathrm{def}^p \leadsto \Theta';\Phi';\Gamma'}\ \text{CHECK\_DEFS\_NIL}$$

$$\frac{\Theta;\Phi;\Gamma \vdash \mathrm{def}^p \leadsto \Phi';\Gamma' \quad \Theta';\Phi';\Gamma' \vdash \mathrm{def}_1^p\,..\,\mathrm{def}_n^p \leadsto \Theta'';\Phi'';\Gamma''}{\Theta;\Phi;\Gamma \vdash \mathrm{def}^p\,\mathrm{def}_1^p\,..\,\mathrm{def}_n^p \leadsto \Theta'';\Phi'';\Gamma''}\ \text{CHECK\_DEFS\_CONS}$$

```
Definition rules:        86 good    0 bad
Definition rule clauses: 242 good    0 bad
```