

<i>ident, x, y, y_p, y_f, -, abbrev, r</i>	subscripts: p for pointers, f for functions
<i>n, i, j</i>	index variables
<i>impl_const</i>	implementation-defined constant
<i>member</i>	C struct/union member name
	Ott-hack, ignore (annotations)
<i>nat</i>	OCaml arbitrary-width natural number
<i>mem_ptr</i>	abstract pointer value
<i>mem_val</i>	abstract memory value
	Ott-hack, ignore (locations)
<i>mem_iv_c</i>	OCaml type for memory constraints on integer values
<i>UB_name</i>	undefined behaviour
<i>string</i>	OCaml string
	Ott-hack, ignore (OCaml type variable TY)
	Ott-hack, ignore (OCaml Symbol.prefix)
<i>mem_order, _</i>	OCaml type for memory order
<i>linux_mem_order</i>	OCaml type for Linux memory order
	Ott-hack, ignore (OCaml type variable bt)

$Stypes_t, \tau$	$::=$	C type
	τ^*	pointer to type τ
tag	$::=$	OCaml type for struct/union tag
	$ident$	
$\beta, -$	$::=$	base types
	unit	unit
	bool	boolean
	integer	integer
	real	rational numbers?
	loc	location
	array β	array
	list β	list
	$\overline{\beta_i}^i$	tuple
	struct tag	struct
	set β	set
	opt (β)	option
	$\beta \rightarrow \beta'$	parameter types
	β_τ M	of a C type
$binop$	$::=$	binary operators
	+	addition
	-	subtraction
	*	multiplication
	/	division
	rem_t	modulus
	rem_f	remainder
	^	exponentiation
	=	equality, defined both for integer and C types

		!=	inequality, similiarly defined
		>	greater than, similarly defined
		<	less than, similarly defined
		>=	greater than or equal to, similarly defined
		<=	less than or equal to, similarly defined
		/\	conjuction
		\/	disjunction
<i>binop_{arith}</i>	::=		arithmetic binary operators
		+	
		-	
		*	
		/	
		rem_t	
		rem_f	
		^	
<i>binop_{rel}</i>	::=		relational binary operators
		=	
		!=	
		>	
		<	
		>=	
		<=	
<i>binop_{bool}</i>	::=		boolean binary operators
		/\	
		\/	
<i>mem_int</i>	::=		memory integer value

	1	M
	0	M
<i>object_value</i>	$::=$ <i>mem_int</i> <i>mem_ptr</i> $\text{array}(\overline{\text{loaded_value}_i}^i)$ $(\text{struct } ident)\{\overline{\text{member}_i:\tau_i = \text{mem_val}_i}^i\}$ $(\text{union } ident)\{\text{.member} = \text{mem_val}\}$	C object values (inhabitants of object types), which can be read/stored integer value pointer value C array value C struct value C union value
<i>loaded_value</i>	$::=$ <i>specified object_value</i>	potentially unspecified C object values specified loaded value
<i>value</i>	$::=$ <i>object_value</i> <i>loaded_value</i> Unit True False $\beta[\overline{\text{value}_i}^i]$ $(\overline{\text{value}_i}^i)$	Core values C object value loaded C object value unit boolean true boolean false list tuple
<i>bool_value</i>	$::=$ True False	Core booleans boolean true boolean false
<i>ctor_val</i>	$::=$ Nil β Cons Tuple	data constructors empty list list cons tuple

		Array	C array
		Specified	non-unspecified loaded value
<i>ctor_expr</i>	::=		data constructors
		Ivmax	max integer value
		Ivmin	min integer value
		Ivsizeof	sizeof value
		Ivalignof	alignof value
		IvCOMPL	bitwise complement
		IvAND	bitwise AND
		IvOR	bitwise OR
		IvXOR	bitwise XOR
		Fvfromint	cast integer to floating value
		Ivfromfloat	cast floating to integer value
<i>name</i>	::=		
		<i>ident</i>	Core identifier
		<i>impl_const</i>	implementation-defined constant
<i>pval</i>	::=		pure values
		<i>ident</i>	Core identifier
		<i>impl_const</i>	implementation-defined constant
		<i>value</i>	Core values
		$\text{constrained}(\overline{mem_iv_c_i}, \overline{pval_i}^i)$	constrained value
		$\text{error}(\text{string}, pval)$	impl-defined static error
		$\text{ctor_val}(\overline{pval_i}^i)$	data constructor application
		$(\text{struct } ident)\{\overline{member_i = pval_i}^i\}$	C struct expression
		$(\text{union } ident)\{member = pval\}$	C union expression
<i>tpval</i>	::=		top-level pure values

		undef <i>UB_name</i>		undefined behaviour
		done <i>pval</i>		pure done
<i>ident_opt_β</i>	::=			type annotated optional identifier
		<i>_.β</i>	binders = {}	
		<i>ident:β</i>	binders = <i>ident</i>	
<i>pattern</i>	::=			
		<i>ident_opt_β</i>	binders = binders(<i>ident_opt_β</i>)	
		<i>ctor_val(⟦pattern_i⟧ⁱ)</i>	binders = binders(<i>pattern_i</i>)	
<i>z</i>	::=			OCaml arbitrary-width integer
		<i>i</i>	M	literal integer
		<i>mem_int</i>	M	
		<i>size_of(τ)</i>	M	size of a C type
		<i>offset_of_{tag}(member)</i>	M	offset of a struct member
		ptr_size	M	size of a pointer
		<i>max_int_τ</i>	M	maximum value of int of type τ
		<i>min_int_τ</i>	M	minimum value of int of type τ
\mathbb{Q} , <i>q</i> , -	::=			OCaml type for rational numbers
		$\frac{int_1}{int_2}$		
<i>lit</i>	::=			
		<i>ident</i>		
		unit		
		<i>bool</i>		
		<i>z</i>		
		\mathbb{Q}		

<i>ident_or_pattern</i>	$::=$ $\begin{array}{l} \textit{ident} \\ \textit{pattern} \end{array}$	$\text{binders} = \textit{ident}$ $\text{binders} = \text{binders}(\textit{pattern})$
<i>bool_op</i>	$::=$ $\begin{array}{l} \neg \textit{term} \\ \textit{term}_1 = \textit{term}_2 \\ \bigwedge (\overline{\textit{term}_i}^i) \\ \bigvee (\overline{\textit{term}_i}^i) \\ \textit{term}_1 \textit{binop}_{\textit{bool}} \textit{term}_2 \\ \text{if } \textit{term}_1 \text{ then } \textit{term}_2 \text{ else } \textit{term}_3 \end{array}$	\mathbf{M}
<i>arith_op</i>	$::=$ $\begin{array}{l} \textit{term}_1 + \textit{term}_2 \\ \textit{term}_1 - \textit{term}_2 \\ \textit{term}_1 \times \textit{term}_2 \\ \textit{term}_1 / \textit{term}_2 \\ \textit{term}_1 \textbf{rem_t} \textit{term}_2 \\ \textit{term}_1 \textbf{rem_f} \textit{term}_2 \\ \textit{term}_1 \wedge \textit{term}_2 \\ \textit{term}_1 \textit{binop}_{\textit{arith}} \textit{term}_2 \end{array}$	\mathbf{M}
<i>cmp_op</i>	$::=$ $\begin{array}{l} \textit{term}_1 < \textit{term}_2 \\ \textit{term}_1 \leq \textit{term}_2 \\ \textit{term}_1 \textit{binop}_{\textit{rel}} \textit{term}_2 \end{array}$	$\begin{array}{l} \text{less than} \\ \text{less than or equal} \end{array}$ \mathbf{M}
<i>list_op</i>	$::=$ $\begin{array}{l} \textbf{nil} \\ \textbf{tl } \textit{term} \end{array}$	

		$term^{(int)}$
$tuple_op$	$::=$	
		$(\overline{term_i}^i)$
		$term^{(int)}$
$pointer_op$	$::=$	
		mem_ptr
		$term_1 +_{ptr} term_2$
		$cast_int_to_ptr\ term$
		$cast_ptr_to_int\ term$
$array_op$	$::=$	
		$term_1[term_2]$
$param_op$	$::=$	
		$ident:\beta.\ term$
		$term(term_1, \dots, term_n)$
$struct_op$	$::=$	
		$term.member$
ct_pred	$::=$	
		$representable(\tau, term)$
		$aligned(\tau, term)$
		$alignedI(term_1, term_2)$
$term, _$	$::=$	
		lit
		$arith_op$

		<i>bool_op</i>		
		<i>cmp_op</i>		
		<i>tuple_op</i>		
		<i>struct_op</i>		
		<i>pointer_op</i>		
		<i>list_op</i>		
		<i>array_op</i>		
		<i>ct_pred</i>		
		<i>param_op</i>		
		<i>(term)</i>	S	parentheses
		$\sigma(\textit{term})$	M	simul-sub σ in <i>term</i>
		<i>pval</i>	M	
<i>pexpr</i>	::=			pure expressions
		<i>pval</i>		pure values
		<i>ctor_expr</i> ($\overline{pval_i}^i$)		data constructor application
		array_shift (<i>pval</i> ₁ , τ , <i>pval</i> ₂)		pointer array shift
		member_shift (<i>pval</i> , <i>ident</i> , <i>member</i>)		pointer struct/union member shift
		not (<i>pval</i>)		boolean not
		<i>pval</i> ₁ <i>binop</i> <i>pval</i> ₂		binary operations
		memberof (<i>ident</i> , <i>member</i> , <i>pval</i>)		C struct/union member access
		<i>name</i> ($\overline{pval_i}^i$)		pure function call
		assert_undef (<i>pval</i> , <i>UB_name</i>)		
		bool_to_integer (<i>pval</i>)		
		conv_int (τ , <i>pval</i>)		
		wrapI (τ , <i>pval</i>)		
<i>tpexpr</i>	::=			top-level pure expressions
		<i>tpval</i>		top-level pure values
		case <i>pval</i> of $\overline{tpexpr_case_branch_i}^i$ end		pattern matching

	let <i>ident_or_pattern</i> = <i>pexpr</i> in <i>texpr</i>	bind binders(<i>ident_or_pattern</i>) in <i>texpr</i>	pure let
	let <i>ident_or_pattern</i> :($y_1:\beta_1. term_1$) = <i>texpr₁</i> in <i>texpr₂</i>	bind binders(<i>ident_or_pattern</i>) in <i>texpr₂</i>	pure let
	if <i>pval</i> then <i>texpr₁</i> else <i>texpr₂</i>		pure if
	$\sigma(texpr)$	M	simul-sub σ in <i>texpr</i>
<i>texpr_case_branch</i>	::=		pure top-level case expression
	<i>pattern</i> \Rightarrow <i>texpr</i>	bind binders(<i>pattern</i>) in <i>texpr</i>	top-level case expression br
<i>m_kill_kind</i>	::=		
	dynamic		
	static τ		
<i>bool</i> , _	::=		OCaml booleans
	true		
	false		
<i>int</i> , _	::=		OCaml fixed-width integer
	<i>i</i>		literal integer
<i>res_term</i>	::=		resource terms
	emp		empty heap
	<i>points_to</i>		single-cell heap
	<i>ident</i>		variable
	$\langle res_term_1, res_term_2 \rangle$		seperating-conjunction pair
	pack (<i>pval</i> , <i>res_term</i>)		packing for existentials
<i>mem_action</i>	::=		memory actions
	create (<i>pval</i> , τ)		
	create_readonly (<i>pval₁</i> , τ , <i>pval₂</i>)		

	<code>alloc (pval₁, pval₂)</code> <code>kill (m_kill_kind, pval, pt)</code> <code>store (bool, τ, pval₁, pval₂, mem_order, pt)</code> <code>load (τ, pval, mem_order, pt)</code> <code>rmw (τ, pval₁, pval₂, pval₃, mem_order₁, mem_order₂)</code> <code>fence (mem_order)</code> <code>cmp_exch_strong (τ, pval₁, pval₂, pval₃, mem_order₁, mem_order₂)</code> <code>cmp_exch_weak (τ, pval₁, pval₂, pval₃, mem_order₁, mem_order₂)</code> <code>linux_fence (linux_mem_order)</code> <code>linux_load (τ, pval, linux_mem_order)</code> <code>linux_store (τ, pval₁, pval₂, linux_mem_order)</code> <code>linux_rmw (τ, pval₁, pval₂, linux_mem_order)</code>	true means store is locking
<i>polarity</i>	::= <code>neg</code>	polarities for memory actions (pos) sequenced by <code>let weak</code> and <code>let strong</code> only sequenced by <code>let strong</code>
<i>pol_mem_action</i>	::= <code>polarity mem_action</code>	memory actions with polarity
<i>mem_op</i>	::= <code>pval₁ binop_{rel} pval₂</code> <code>pval₁ -_{τ} pval₂</code> <code>intFromPtr (τ_1, τ_2, pval)</code> <code>ptrFromInt (τ_1, τ_2, pval)</code> <code>ptrValidForDeref (τ, pval, pt)</code> <code>ptrWellAligned (τ, pval)</code> <code>ptrArrayShift (pval₁, τ, pval₂)</code> <code>memcpy (pval₁, pval₂, pval₃)</code> <code>memcmp (pval₁, pval₂, pval₃)</code>	operations involving the memory state pointer relational binary operations pointer subtraction cast of pointer value to integer value cast of integer value to pointer value dereferencing validity predicate

		<code>realloc</code> ($pval_1, pval_2, pval_3$)	
		<code>va_start</code> ($pval_1, pval_2$)	
		<code>va_copy</code> ($pval$)	
		<code>va_arg</code> ($pval, \tau$)	
		<code>va_end</code> ($pval$)	
$spine_elem$	$::=$		spine element
		$pval$	pure or logical value
		res_term	resource value
$spine$	$::=$		spine
		$\overline{spine_elem_i}^i$	
$tval$	$::=$		(effectful) top-level values
		<code>done</code> $spine$	end of top-level expression
		<code>undef</code> UB_name	undefined behaviour
$res_pattern$	$::=$		resource terms
		<code>emp</code>	empty heap
		pt	single-cell heap
		$ident$	variable
		$\langle res_pattern_1, res_pattern_2 \rangle$	seperating-conjunction pair
		<code>pack</code> ($ident, res_pattern$)	packing for existentials
		$binders = \{\}$	
		$binders = \{\}$	
		$binders = ident$	
		$binders = binders(res_pattern_1) \cup binders(res_pattern_2)$	
		$binders = ident \cup binders(res_pattern)$	
$ret_pattern$	$::=$		return pattern
		<code>comp</code> $ident_or_pattern$	computational variable
		<code>log</code> $ident$	logical variable
		<code>res</code> $res_pattern$	resource variable
		$binders = binders(ident_or_pattern)$	
		$binders = ident$	
		$binders = binders(res_pattern)$	
$init,$	$::=$		initialisation status

		✓	initialised
		×	uninitialised
$points_to, pt$	$::=$		points-to separation logic predicate
		$term_1 \xrightarrow{init}_\tau term_2$	
res	$::=$		resources
		emp	empty heap
		$points_to$	points-top heap pred.
		$res_1 * res_2$	seperating conjunction
		$\exists ident:\beta. res$	existential
		$term \wedge res$	logical conjunction
		$\langle res \rangle$	S parentheses
		$[\sigma]res$	M simul-sub σ in res
$ret, -$	$::=$		return types
		$\Sigma ident:\beta. ret$	return a computational value
		$\exists ident:\beta. ret$	return a logical value
		$res \otimes ret$	return a resource value
		$term \wedge ret$	return a predicate (post-condition)
		I	end return list
		$\sigma(ret)$	M simul-sub σ in ret
seq_expr	$::=$		sequential (effectful) expressions
		ccall $(\tau, pval, spine)$	C function call
		pcall $(name, spine)$	procedure call
seq_texpr	$::=$		sequential top-level (effectful) expressions
		$tval$	(effectful) top-level values
		run $ident \overline{pval}_i^i$	run from label

	let $ident_or_pattern = pexpr$ in $texpr$	bind $binders(ident_or_pattern)$ in $texpr$	pure let
	let $ident_or_pattern:(y_1:\beta_1. term_1) = tpexpr$ in $texpr$	bind $binders(ident_or_pattern)$ in $texpr$	pure let
	let $\overline{ret_pattern_i}^i = seq_expr$ in $texpr$	bind $binders(\overline{ret_pattern_i}^i)$ in $texpr$	bind return patterns
	let $\overline{ret_pattern_i}^i:ret = texpr_1$ in $texpr_2$	bind $binders(\overline{ret_pattern_i}^i)$ in $texpr_2$	annotated bind return patterns
	case $pval$ of $texpr_case_branch_i$ end		pattern matching
	if $pval$ then $texpr_1$ else $texpr_2$		conditional
	bound $[int](is_texpr)$		limit scope of indet seq behaviour
$texpr_case_branch$	$::=$		top-level case expression branch
	$pattern \Rightarrow texpr$	bind $binders(pattern)$ in $texpr$	top-level case expression branch
is_expr	$::=$		indet seq (effectful) expressions
	$tval$		(effectful) top-level values
	memop (mem_op)		pointer op involving memory
	pol_mem_action		memory action
is_texpr	$::=$		indet seq top-level (effectful) expressions
	let weak $\overline{ret_pattern_i}^i = is_expr$ in $texpr$	bind $binders(\overline{ret_pattern_i}^i)$ in $texpr$	weak sequencing
	let strong $\overline{ret_pattern_i}^i = is_expr$ in $texpr$	bind $binders(\overline{ret_pattern_i}^i)$ in $texpr$	strong sequencing
$texpr$	$::=$		top-level (effectful) expressions
	seq_texpr		sequential (effectful) expressions
	is_texpr		indet seq (effectful) expressions
	$\sigma(texpr)$	M	simul-sub σ in $texpr$
arg	$::=$		argument/function types
	$\Pi ident:\beta. arg$		
	$\forall ident:\beta. arg$		
	$res \multimap arg$		

	$ \begin{array}{l} \quad term \supset arg \\ \quad ret \\ \quad \sigma(arg) \end{array} $	M	simul-sub σ in arg
$pure_arg$	$ \begin{array}{l} ::= \\ \quad \Pi ident:\beta. pure_arg \\ \quad term \supset pure_arg \\ \quad pure_ret \end{array} $		pure argument/function types
$pure_ret$	$ \begin{array}{l} ::= \\ \quad \Sigma ident:\beta. pure_ret \\ \quad term \wedge pure_ret \\ \quad \mathbf{I} \end{array} $		pure return types
\mathcal{C}	$ \begin{array}{l} ::= \\ \quad . \\ \quad \mathcal{C}, ident:\beta \\ \quad \overline{\mathcal{C}}_i^i \end{array} $		computational var env
\mathcal{L}	$ \begin{array}{l} ::= \\ \quad . \\ \quad \overline{\mathcal{L}}_i^i \\ \quad \mathcal{L}, ident:\beta \end{array} $		logical var env
Φ	$ \begin{array}{l} ::= \\ \quad . \\ \quad \Phi, term \\ \quad \overline{\Phi}_i^i \end{array} $		constraints env
\mathcal{R}	$::=$		resources env

	$\begin{array}{ l} \cdot \\ \mathcal{R}, ident:res \\ \overline{\mathcal{R}}_i^i \end{array}$	
σ	$\begin{array}{ l} ::= \\ \cdot \\ spine_elem/ident, \sigma \\ \overline{\sigma}_i^i \end{array}$	substitutions
<i>typing</i>	$\begin{array}{ l} ::= \\ \mathbf{smt}(\Phi \Rightarrow term) \\ ident:\beta \in \mathcal{C} \\ ident:\beta \in \mathcal{L} \\ ident:\mathbf{struct\ tag} \ \& \ \overline{member_i:\tau_i}^i \in \mathbf{Globals} \\ \overline{\mathcal{C}_i;\mathcal{L}_i;\Phi_i} \vdash mem_val_i \Rightarrow \mathbf{mem} \ \overline{\beta_i}^i \end{array}$	dependent on memory object model
<i>opsem</i>	$\begin{array}{ l} ::= \\ \forall i < j. \mathbf{not} (pattern_i = pval \rightsquigarrow \sigma_i) \\ \mathbf{fresh}(mem_ptr) \\ term \\ pval:\beta \end{array}$	
<i>formula</i>	$\begin{array}{ l} ::= \\ judgement \\ typing \\ opsem \\ term \equiv term' \\ name: pure_arg \equiv \overline{x_i}^i \mapsto texpr \in \mathbf{Globals} \\ pval: arg \equiv \overline{x_i}^i \mapsto texpr \in \mathbf{Globals} \end{array}$	

$heap, h$	$::=$ $\begin{array}{ l} \cdot \\ h + \{points_to\} \end{array}$	heaps
$object_value_jtype$	$::=$ $\begin{array}{ l} \mathcal{C}; \mathcal{L}; \Phi \vdash object_value \Rightarrow \mathbf{obj} \beta \end{array}$	
$pval_jtype$	$::=$ $\begin{array}{ l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \end{array}$	
res_jtype	$::=$ $\begin{array}{ l} \Phi \vdash res \equiv res' \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow res \end{array}$	
$spine_jtype$	$::=$ $\begin{array}{ l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}$	
$pexpr_jtype$	$::=$ $\begin{array}{ l} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow ident:\beta. term \end{array}$	
$comp_pattern_jtype$	$::=$ $\begin{array}{ l} term \mathbf{as} pattern:\beta \rightsquigarrow \mathcal{C}; \Phi \\ term \mathbf{as} ident_or_pattern:\beta \rightsquigarrow \mathcal{C}; \Phi \end{array}$	
$res_pattern_jtype$	$::=$ $\begin{array}{ l} res_pattern:res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R} \end{array}$	
$ret_pattern_jtype$	$::=$ $\begin{array}{ l} \overline{ret_pattern_i}^i : ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \end{array}$	

$tpval_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi \vdash tpval \Leftarrow ident:\beta. term$
$tpexpr_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr \Leftarrow ident:\beta. term$
$action_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret$
$memop_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_op \Rightarrow ret$
$tval_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret$
seq_expr_jtype	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_expr \Rightarrow ret$
is_expr_jtype	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is_expr \Rightarrow ret$
$texpr_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_texpr \Leftarrow ret$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is_texpr \Leftarrow ret$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash texpr \Leftarrow ret$
$subs_jtype$	$::=$	$pattern = pval \rightsquigarrow \sigma$ $ident_or_pattern = pval \rightsquigarrow \sigma$ $res_pattern = res_term \rightsquigarrow \sigma$

		$\frac{ret_pattern = spine_elem \rightsquigarrow \sigma}{x_i = spine_elem_i^i :: arg \gg \sigma; ret}$
$pure_opsem_jtype$	$::=$	$\langle pexpr \rangle \longrightarrow \langle pexpr' \rangle$ $\langle pexpr \rangle \longrightarrow \langle texpr:(y:\beta. term) \rangle$ $\langle texpr \rangle \longrightarrow \langle texpr' \rangle$
$opsem_jtype$	$::=$	$\langle seq_expr \rangle \longrightarrow \langle texpr:ret \rangle$ $\langle h; seq_texpr \rangle \longrightarrow \langle h'; texpr \rangle$ $\langle h; mem_op \rangle \longrightarrow \langle h'; tval \rangle$ $\langle h; mem_action \rangle \longrightarrow \langle h'; tval \rangle$ $\langle h; is_expr \rangle \longrightarrow \langle h'; is_expr' \rangle$ $\langle h; is_texpr \rangle \longrightarrow \langle h'; texpr \rangle$ $\langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle$
$lemma_jtype$	$::=$	$\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash object_value \Rightarrow \mathbf{obj} \beta}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash mem_int \Rightarrow \mathbf{obj} \mathbf{integer}} \quad \text{TY_PVAL_OBJ_INT}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash mem_ptr \Rightarrow \mathbf{obj} \mathbf{loc}} \quad \text{TY_PVAL_OBJ_PTR}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash loaded_value_i \Rightarrow \beta^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathbf{array}(\overline{loaded_value_i}^i) \Rightarrow \mathbf{obj} \mathbf{array} \beta} \quad \text{TY_PVAL_OBJ_ARR}$$

$$\frac{\frac{\text{ident}:\text{struct tag} \ \& \ \overline{\text{member}_i:\tau_i}^i \in \text{Globals}}{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{mem_val}_i \Rightarrow \text{mem } \beta_{\tau_i}}^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash (\text{struct tag})\{.\text{member}_i:\tau_i = \text{mem_val}_i\} \Rightarrow \text{obj struct tag}} \quad \text{TY_PVAL_OBJ_STRUCT}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{pval} \Rightarrow \beta}$$

$$\frac{x:\beta \in \mathcal{C}}{\mathcal{C}; \mathcal{L}; \Phi \vdash x \Rightarrow \beta} \quad \text{TY_PVAL_VAR_COMP}$$

$$\frac{x:\beta \in \mathcal{L}}{\mathcal{C}; \mathcal{L}; \Phi \vdash x \Rightarrow \beta} \quad \text{TY_PVAL_VAR_LOG}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{object_value} \Rightarrow \text{obj } \beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{object_value} \Rightarrow \beta} \quad \text{TY_PVAL_OBJ}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{object_value} \Rightarrow \text{obj } \beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{specified object_value} \Rightarrow \beta} \quad \text{TY_PVAL_LOADED}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Unit} \Rightarrow \text{unit}} \quad \text{TY_PVAL_UNIT}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{True} \Rightarrow \text{bool}} \quad \text{TY_PVAL_TRUE}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{False} \Rightarrow \text{bool}} \quad \text{TY_PVAL_FALSE}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{value}_i \Rightarrow \beta^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \beta[\overline{\text{value}_i}^i] \Rightarrow \text{list } \beta} \quad \text{TY_PVAL_LIST}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash value_i \Rightarrow \beta_i^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash (\overline{value_i^i}) \Rightarrow \overline{\beta_i^i}} \quad \text{TY_PVAL_TUPLE}$$

$$\frac{\text{smt}(\Phi \Rightarrow \text{false})}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{error}(string, pval) \Rightarrow \beta} \quad \text{TY_PVAL_ERROR}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Nil } \beta() \Rightarrow \text{list } \beta} \quad \text{TY_PVAL_CTOR_NIL}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{list } \beta \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Cons}(pval_1, pval_2) \Rightarrow \text{list } \beta} \quad \text{TY_PVAL_CTOR_CONS}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta_i^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Tuple}(\overline{pval_i^i}) \Rightarrow \overline{\beta_i^i}} \quad \text{TY_PVAL_CTOR_TUPLE}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta^i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Array}(\overline{pval_i^i}) \Rightarrow \text{array } \beta} \quad \text{TY_PVAL_CTOR_ARRAY}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{Specified}(pval) \Rightarrow \beta} \quad \text{TY_PVAL_CTOR_SPECIFIED}$$

$$\frac{\begin{array}{l} ident: \text{struct tag} \ \& \ \overline{member_i: \tau_i^i} \in \text{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta_{\tau_i^i} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash (\text{struct tag})\{.member_i = pval_i^i\} \Rightarrow \text{struct tag}} \quad \text{TY_PVAL_STRUCT}$$

$$\boxed{\Phi \vdash res \equiv res'}$$

$$\frac{}{\Phi \vdash \text{emp} \equiv \text{emp}} \quad \text{TYRES_EQ_EMP}$$

$$\frac{\text{smt}(\Phi \Rightarrow (term_1 = term'_1) \wedge (term_2 = term'_2))}{\Phi \vdash term_1 \xrightarrow{\text{init}}_{\tau} term_2 \equiv term'_1 \xrightarrow{\text{init}}_{\tau} term'_2} \quad \text{TYRES_EQ_POINTSTO}$$

$$\frac{\begin{array}{l} \Phi \vdash res_1 \equiv res'_1 \\ \Phi \vdash res_2 \equiv res'_2 \end{array}}{\Phi \vdash res_1 * res_2 \equiv res'_1 * res'_2} \quad \text{TYRES_EQ_SEP_CONJ}$$

$$\frac{\Phi \vdash res \equiv res'}{\Phi \vdash \exists ident:\beta. res \equiv \exists ident:\beta. res'} \quad \text{TYRES_EQ_EXISTS}$$

$$\frac{\begin{array}{l} \text{smt}(\Phi, term \Rightarrow term') \\ \text{smt}(\Phi, term' \Rightarrow term) \\ \Phi \vdash res \equiv res' \end{array}}{\Phi \vdash term \wedge res \equiv term' \wedge res'} \quad \text{TYRES_EQ_TERM}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow res}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{emp} \Leftarrow \text{emp}} \quad \text{TYRES_EMP}$$

$$\frac{\Phi \vdash points_to \equiv points_to'}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, \cdot : points_to \vdash points_to' \Leftarrow points_to'} \quad \text{TYRES_POINTSTO}$$

$$\frac{\Phi \vdash res \equiv res'}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, r : res \vdash r \Leftarrow res'} \quad \text{TYRES_VAR}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res_term_1 \Leftarrow res_1 \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash res_term_2 \Leftarrow res_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \langle res_term_1, res_term_2 \rangle \Leftarrow res_1 * res_2} \text{TYRES_SEP_CONJ}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term_2 \Leftarrow [pval/y, \cdot]res \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{pack}(pval, res_term_2) \Leftarrow \exists y:\beta. res} \text{TYRES_PACK}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash :: ret \gg \cdot; ret} \text{TY_SPINE_EMPTY}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash x = pval, \overline{x_i = spine_elem_i}^i :: \Pi x:\beta. arg \gg pval/x, \sigma; ret} \text{TY_SPINE_COMP}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash x = pval, \overline{x_i = spine_elem_i}^i :: \forall x:\beta. arg \gg pval/x, \sigma; ret} \text{TY_SPINE_LOG}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res_term \Leftarrow res \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash x = res_term, \overline{x_i = spine_elem_i}^i :: res \multimap arg \gg res_term/x, \sigma; ret} \text{TY_SPINE_RES}$$

$$\frac{\begin{array}{c} \text{smt}(\Phi \Rightarrow term) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: term \supset arg \gg \sigma; ret} \text{TY_SPINE_PHI}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow ident:\beta.term}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow y:\beta. y = pval} \quad \text{TY_PE_VAL}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{loc} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{integer} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{array_shift}(pval_1, \tau, pval_2) \Rightarrow y:\text{loc}. y = pval_1 +_{\text{ptr}} (pval_2 \times \text{size_of}(\tau))} \quad \text{TY_PE_ARRAY_SHIFT}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{loc} \\ \therefore \text{struct tag} \ \& \ \overline{\text{member}_i:\tau_i}^i \in \text{Globals} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{member_shift}(pval, \text{tag}, \text{member}_j) \Rightarrow y:\text{loc}. y = pval +_{\text{ptr}} \text{offset_of}_{\text{tag}}(\text{member}_j)} \quad \text{TY_PE_MEMBER_SHIFT}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{not}(pval) \Rightarrow y:\text{bool}. y = \neg pval} \quad \text{TY_PE_NOT}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{integer} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{integer} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \text{ binop}_{\text{arith}} pval_2 \Rightarrow y:\text{integer}. y = (pval_1 \text{ binop}_{\text{arith}} pval_2)} \quad \text{TY_PE_ARITH_BINOP}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{integer} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{integer} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \text{ binop}_{\text{rel}} pval_2 \Rightarrow y:\text{bool}. y = (pval_1 \text{ binop}_{\text{rel}} pval_2)} \quad \text{TY_PE_REL_BINOP}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{bool} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{bool} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \text{ binop}_{\text{bool}} pval_2 \Rightarrow y:\text{bool}. y = (pval_1 \text{ binop}_{\text{bool}} pval_2)} \quad \text{TY_PE_BOOL_BINOP}$$

$$\frac{\begin{array}{l} name: pure_arg \equiv \overline{x_i}^i \mapsto tpepr \in \mathbf{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \overline{x_i = pval_i}^i :: pure_arg \gg \sigma; \Sigma y:\beta. term \wedge \mathbf{I} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash name(\overline{pval_i}^i) \Rightarrow y:\beta. \sigma(term)} \quad \text{TY_PE_CALL}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \mathbf{bool} \\ \mathbf{smt}(\Phi \Rightarrow pval) \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathbf{assert_undef}(pval, UB_name) \Rightarrow y:\mathbf{unit}. y = \mathbf{unit}} \quad \text{TY_PE_ASSERT_UNDEF}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \mathbf{bool}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathbf{bool_to_integer}(pval) \Rightarrow y:\mathbf{integer}. y = \mathbf{if } pval \mathbf{ then } 1 \mathbf{ else } 0} \quad \text{TY_PE_BOOL_TO_INTEGER}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \mathbf{integer} \\ abbrev_1 \equiv \max_int_\tau - \min_int_\tau + 1 \\ abbrev_2 \equiv pval \mathbf{rem_f} abbrev_1 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathbf{wrapI}(\tau, pval) \Rightarrow y:\beta. y = \mathbf{if } abbrev_2 \leq \max_int_\tau \mathbf{ then } abbrev_2 \mathbf{ else } abbrev_2 - abbrev_1} \quad \text{TY_PE_WRAP I}$$

$term \mathbf{as} pattern:\beta \rightsquigarrow \mathcal{C}; \Phi$

$$\frac{}{term \mathbf{as} _:\beta:\beta \rightsquigarrow \cdot; \cdot} \quad \text{TY_PAT_COMP_NO_SYM_ANNOT}$$

$$\frac{}{term \mathbf{as} x:\beta:\beta \rightsquigarrow \cdot, x:\beta; \cdot, x = term} \quad \text{TY_PAT_COMP_SYM_ANNOT}$$

$$\frac{}{term \mathbf{as Nil } \beta():\mathbf{list } \beta \rightsquigarrow \cdot; \cdot} \quad \text{TY_PAT_COMP_NIL}$$

$$\frac{\begin{array}{l} term^{(1)} \mathbf{as} pattern_1:\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathbf{tl } term \mathbf{as} pattern_2:\mathbf{list } \beta \rightsquigarrow \mathcal{C}_2; \Phi_1 \end{array}}{term \mathbf{as Cons}(pattern_1, pattern_2):\mathbf{list } \beta \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \Phi_1, \Phi_2} \quad \text{TY_PAT_COMP_CONS}$$

$$\frac{\overline{term^{(i)} \text{ as } pattern_i : \beta_i \rightsquigarrow \mathcal{C}_i; \Phi_i^i}}{term \text{ as Tuple}(\overline{pattern_i^i}) : \overline{\beta_i^i \rightsquigarrow \mathcal{C}_i^i; \Phi_i^i}} \quad \text{TY_PAT_COMP_TUPLE}$$

$$\frac{\overline{term[i] \text{ as } pattern_i : \beta \rightsquigarrow \mathcal{C}_i; \Phi_i^i}}{term \text{ as Array}(\overline{pattern_i^i}) : \text{array } \beta \rightsquigarrow \overline{\mathcal{C}_i^i; \Phi_i^i}} \quad \text{TY_PAT_COMP_ARRAY}$$

$$\frac{term \text{ as } pattern : \beta \rightsquigarrow \mathcal{C}; \Phi}{term \text{ as Specified}(pattern) : \beta \rightsquigarrow \mathcal{C}; \Phi} \quad \text{TY_PAT_COMP_SPECIFIED}$$

$$\boxed{term \text{ as ident_or_pattern} : \beta \rightsquigarrow \mathcal{C}; \Phi}$$

$$\frac{}{term \text{ as } x : \beta \rightsquigarrow \cdot, x : \beta; \cdot, x = term} \quad \text{TY_PAT_SYM_OR_PATTERN_SYM}$$

$$\frac{term \text{ as } pattern : \beta \rightsquigarrow \mathcal{C}; \Phi}{term \text{ as } pattern : \beta \rightsquigarrow \mathcal{C}; \Phi} \quad \text{TY_PAT_SYM_OR_PATTERN_PATTERN}$$

$$\boxed{res_pattern : res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}$$

$$\frac{}{\text{emp} : \text{emp} \rightsquigarrow \cdot; \cdot; \cdot} \quad \text{TY_PAT_RES_EMPTY}$$

$$\frac{}{points_to : points_to \rightsquigarrow \cdot; \cdot; \cdot, \cdot : points_to} \quad \text{TY_PAT_RES_POINTSTO}$$

$$\frac{}{r : res \rightsquigarrow \cdot; \cdot; \cdot, r : res} \quad \text{TY_PAT_RES_VAR}$$

$$\frac{\begin{array}{l} res_pattern_1 : res_1 \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ res_pattern_2 : res_2 \rightsquigarrow \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\langle res_pattern_1, res_pattern_2 \rangle : res_1 * res_2 \rightsquigarrow \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2} \quad \text{TY_PAT_RES_SEPCONJ}$$

$$\frac{res_pattern:res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}{res_pattern:term \wedge res \rightsquigarrow \mathcal{L}; \Phi, term; \mathcal{R}} \quad \text{TY_PAT_RES_CONJ}$$

$$\frac{res_pattern:[x/y, \cdot]res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}{\text{pack}(x, res_pattern):\exists y:\beta. res \rightsquigarrow \mathcal{L}, x:\beta; \Phi; \mathcal{R}} \quad \text{TY_PAT_RES_PACK}$$

$$\boxed{\overline{ret_pattern_i}^i:ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}$$

$$\frac{}{:I \rightsquigarrow \cdot; \cdot; \cdot; \cdot} \quad \text{TY_PAT_RET_EMPTY}$$

$$\frac{\frac{y \text{ as } ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1; \Phi_1}{\overline{ret_pattern_i}^i:ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2}}{\text{comp } ident_or_pattern, \overline{ret_pattern_i}^i:\Sigma y:\beta. ret \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2} \quad \text{TY_PAT_RET_COMP}$$

$$\frac{\overline{ret_pattern_i}^i:ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\log y, \overline{ret_pattern_i}^i:\exists y:\beta. ret \rightsquigarrow \mathcal{C}; \mathcal{L}, y:\beta; \Phi; \mathcal{R}} \quad \text{TY_PAT_RET_LOG}$$

$$\frac{\frac{res_pattern:res \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1}{\overline{ret_pattern_i}^i:ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2}}{res\ res_pattern, \overline{ret_pattern_i}^i:res \otimes ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2} \quad \text{TY_PAT_RET_RES}$$

$$\frac{\overline{ret_pattern_i}^i:ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\overline{ret_pattern_i}^i:term \wedge ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi, term; \mathcal{R}} \quad \text{TY_PAT_RET_PHI}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash tpval \Leftarrow ident:\beta. term}$$

$$\frac{\text{smt}(\Phi \Rightarrow \text{false})}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{undef } UB_name \Leftarrow y:\beta. term} \quad \text{TY_TPVAL_UNDEF}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \quad \text{smt}(\Phi \Rightarrow pval/y, \cdot(term))}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{done } pval \Leftarrow y:\beta. term} \quad \text{TY_TPVAL_DONE}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash tpepr \Leftarrow ident:\beta. term}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool} \\ \mathcal{C}; \mathcal{L}; \Phi, pval = \text{true} \vdash tpepr_1 \Leftarrow y:\beta. term \\ \mathcal{C}; \mathcal{L}; \Phi, pval = \text{false} \vdash tpepr_2 \Leftarrow y:\beta. term \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{if } pval \text{ then } tpepr_1 \text{ else } tpepr_2 \Leftarrow y:\beta. term} \quad \text{TY_TPE_IF}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y_1:\beta_1. term_1 \\ y_1 \text{ as } ident_or_pattern:\beta_1 \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, y_1:\beta_1; \Phi, term_1, \Phi_1 \vdash tpepr \Leftarrow y_2:\beta_2. term_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{let } ident_or_pattern = pexpr \text{ in } tpepr \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_LET}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash tpepr_1 \Leftarrow y_1:\beta_1. term_1 \\ y_1 \text{ as } ident_or_pattern:\beta_1 \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, y_1:\beta_1; \Phi, term_1, \Phi_1 \vdash tpepr \Leftarrow y_2:\beta_2. term_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{let } ident_or_pattern:(y_1:\beta_1. term_1) = tpepr_1 \text{ in } tpepr_2 \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_LETT}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta_1 \\ \overline{y_1 \text{ as } pattern_i:\beta_1 \rightsquigarrow \mathcal{C}_i; \Phi_i}^i \\ \mathcal{C}, \mathcal{C}_i; \mathcal{L}, y_1:\beta_1; \Phi, y_1 = pval, \Phi_i \vdash tpepr_i \Leftarrow y_2:\beta_2. term_2^i \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{case } pval \text{ of } \overline{pattern_i \Rightarrow tpepr_i}^i \text{ end} \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_CASE}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{integer}}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{create}(pval, \tau) \Rightarrow \Sigma y_p:\text{loc. representable}(\tau*, y_p) \wedge \text{alignedI}(pval, y_p) \wedge \exists y:\beta_\tau. \langle y_p \xrightarrow{\times}_\tau y \rangle \otimes \mathbf{I}} \quad \text{TY_ACTION_CREATE}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \text{loc} \\
\text{smt}(\Phi \Rightarrow pval_0 = pval_1) \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_1 \overset{\checkmark}{\mapsto}_{\tau} pval_2 \Leftarrow pval_1 \overset{\checkmark}{\mapsto}_{\tau} pval_2 \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{load}(\tau, pval_0, -, pval_1 \overset{\checkmark}{\mapsto}_{\tau} pval_2) \Rightarrow \Sigma y:\beta_{\tau}. y = pval_2 \wedge \langle pval_1 \overset{\checkmark}{\mapsto}_{\tau} pval_2 \rangle \otimes \mathbf{I}
\end{array}
\quad \text{TY_ACTION_LOAD}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \text{loc} \\
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \beta_{\tau} \\
\text{smt}(\Phi \Rightarrow \text{representable}(\tau, pval_1)) \\
\text{smt}(\Phi \Rightarrow pval_2 = pval_0) \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_2 \mapsto_{\tau} - \Leftarrow pval_2 \mapsto_{\tau} - \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{store}(-, \tau, pval_0, pval_1, -, pval_2 \mapsto_{\tau} -) \Rightarrow \Sigma _:\text{unit}. \langle pval_2 \overset{\checkmark}{\mapsto}_{\tau} pval_1 \rangle \otimes \mathbf{I}
\end{array}
\quad \text{TY_ACTION_STORE}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \text{loc} \\
\text{smt}(\Phi \Rightarrow pval_0 = pval_1) \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_1 \mapsto_{\tau} - \Leftarrow pval_1 \mapsto_{\tau} - \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{kill}(\text{static } \tau, pval_0, pval_1 \mapsto_{\tau} -) \Rightarrow \Sigma _:\text{unit}. \mathbf{I}
\end{array}
\quad \text{TY_ACTION_KILL_STATIC}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{mem_op} \Rightarrow \text{ret}}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{loc} \\
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{loc} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash pval_1 \text{ binop}_{rel} pval_2 \Rightarrow \Sigma y:\text{bool}. y = (pval_1 \text{ binop}_{rel} pval_2) \wedge \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_REL_BINOP}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{loc} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{intFromPtr}(\tau_1, \tau_2, pval) \Rightarrow \Sigma y:\text{integer}. y = \text{cast_ptr_to_int } pval \wedge \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_INTFROMPTR}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{integer} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{ptrFromInt}(\tau_1, \tau_2, pval) \Rightarrow \Sigma y:\text{loc}. y = \text{cast_int_to_ptr } pval \wedge \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_PTRFROMINT}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \text{loc} \\
\text{smt}(\Phi \Rightarrow pval_1 = pval_0) \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_1 \overset{\checkmark}{\mapsto}_{\tau} - \Leftarrow pval_1 \overset{\checkmark}{\mapsto}_{\tau} - \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{ptrValidForDeref}(\tau, pval_0, pval_1 \overset{\checkmark}{\mapsto}_{\tau} -) \Rightarrow \Sigma y:\text{bool}. y = \text{aligned}(\tau, pval_1) \wedge \langle pval_1 \overset{\checkmark}{\mapsto}_{\tau} - \rangle \otimes \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_PTRVALIDFORDEREF}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{loc} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{ptrWellAligned}(\tau, pval) \Rightarrow \Sigma y:\text{bool}. y = \text{aligned}(\tau, pval) \wedge \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_PTRWELLALIGNED}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \text{loc} \\
\mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \text{integer} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{ptrArrayShift}(pval_1, \tau, pval_2) \Rightarrow \Sigma y:\text{loc}. y = pval_1 +_{\text{ptr}} (pval_2 \times \text{size_of}(\tau)) \wedge \mathbf{I}
\end{array}
\quad \text{TY_MEMOP_PTRARRAYSHIFT}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow \text{ret}}$$

$$\overline{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{done} \Leftarrow \mathbf{I}} \quad \text{TY_TVAL_I}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done } \overline{\text{spine_elem}_i}^i \Leftarrow pval/y, \cdot(\text{ret}) \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done } pval, \overline{\text{spine_elem}_i}^i \Leftarrow \Sigma y:\beta. \text{ret}
\end{array}
\quad \text{TY_TVAL_COMP}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done } \overline{\text{spine_elem}_i}^i \Leftarrow pval/y, \cdot(\text{ret}) \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done } pval, \overline{\text{spine_elem}_i}^i \Leftarrow \exists y:\beta. \text{ret}
\end{array}
\quad \text{TY_TVAL_LOG}$$

$$\begin{array}{c}
\text{smt}(\Phi \Rightarrow \text{term}) \\
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done spine} \Leftarrow \text{ret} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{done spine} \Leftarrow \text{term} \wedge \text{ret}
\end{array}
\quad \text{TY_TVAL_PHI}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res_term \Leftarrow res \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash \text{done } \overline{spine_elem_i}^i \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \text{done } res_term, \overline{spine_elem}^i \Leftarrow res \otimes ret} \quad \text{TY_TVAL_RES}$$

$$\frac{\text{smt}(\Phi \Rightarrow \text{false})}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{undef } UB_name \Leftarrow ret} \quad \text{TY_TVAL_UB}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_expr \Rightarrow ret}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{loc} \\ pval:arg \equiv \overline{x_i}^i \mapsto texpr \in \text{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{ccall}(\tau, pval, \overline{spine_elem_i}^i) \Rightarrow \sigma(ret)} \quad \text{TY_SEQ_E_CCALL}$$

$$\frac{\begin{array}{c} name:arg \equiv \overline{x_i}^i \mapsto texpr \in \text{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{pcall}(name, \overline{spine_elem_i}^i) \Rightarrow \sigma(ret)} \quad \text{TY_SEQ_E_PROC}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is_expr \Rightarrow ret}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_op \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{memop}(mem_op) \Rightarrow ret} \quad \text{TY_IS_E_MEMOP}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret} \quad \text{TY_IS_E_ACTION}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{neg } mem_action \Rightarrow ret} \quad \text{TY_IS_E_NEG_ACTION}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_expr \Leftarrow ret}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret} \text{TY_SEQ_TE_TVAL}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y:\beta. term \\ y \text{ as } ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, y:\beta; \Phi, term, \Phi_1; \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{let } ident_or_pattern = pexpr \text{ in } texpr \Leftarrow ret} \text{TY_SEQ_TE_LETP}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr \Leftarrow y:\beta. term \\ y \text{ as } ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, y:\beta; \Phi, term, \Phi_1; \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{let } ident_or_pattern:(y:\beta. term) = tpexpr \text{ in } texpr \Leftarrow ret} \text{TY_SEQ_TE_LETPPT}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}' \vdash seq_expr \Rightarrow ret_1 \\ \overline{ret_pattern_i}^i : ret_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, \mathcal{L}_1; \Phi, \Phi_1; \mathcal{R}, \mathcal{R}_1 \vdash texpr \Leftarrow ret_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}', \mathcal{R} \vdash \text{let } \overline{ret_pattern_i}^i = seq_expr \text{ in } texpr \Leftarrow ret_2} \text{TY_SEQ_TE_LET}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}' \vdash texpr_1 \Leftarrow ret_1 \\ \overline{ret_pattern_i}^i : ret_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, \mathcal{L}_1; \Phi, \Phi_1; \mathcal{R}, \mathcal{R}_1 \vdash texpr_2 \Leftarrow ret_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}', \mathcal{R} \vdash \text{let } \overline{ret_pattern_i}^i : ret_1 = texpr_1 \text{ in } texpr_2 \Leftarrow ret_2} \text{TY_SEQ_TE_LETT}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta_1 \\ \overline{y_1 \text{ as } pattern_i:\beta_1 \rightsquigarrow \mathcal{C}_i;\Phi_i}^i \\ \overline{\mathcal{C}, \mathcal{C}_i; \mathcal{L}, y_1:\beta_1; \Phi, y_1 = pval, \Phi_i; \mathcal{R} \vdash texpr_i \Leftarrow ret}^i \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{case } pval \text{ of } \overline{pattern_i \Rightarrow texpr_i}^i \text{ end} \Leftarrow ret} \text{TY_SEQ_TE_CASE}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool} \\
\mathcal{C}; \mathcal{L}; \Phi, pval = \text{true}; \mathcal{R} \vdash \text{expr}_1 \Leftarrow \text{ret} \\
\mathcal{C}; \mathcal{L}; \Phi, pval = \text{false}; \mathcal{R} \vdash \text{expr}_2 \Leftarrow \text{ret} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{if } pval \text{ then } \text{expr}_1 \text{ else } \text{expr}_2 \Leftarrow \text{ret}
\end{array}
\quad \text{TY_SEQ_TE_IF}$$

$$\begin{array}{c}
\text{ident} : \text{arg} \equiv \overline{x_i}^i \mapsto \text{expr} \in \text{Globals} \\
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \overline{x_i = pval_i}^i :: \text{arg} \gg \sigma; \text{false} \wedge \text{I} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{run ident } \overline{pval_i}^i \Leftarrow \text{false} \wedge \text{I}
\end{array}
\quad \text{TY_SEQ_TE_RUN}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_expr} \Leftarrow \text{ret} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{bound}[\text{int}](\text{is_expr}) \Leftarrow \text{ret}
\end{array}
\quad \text{TY_SEQ_TE_BOUND}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_expr} \Leftarrow \text{ret}}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_expr} \Rightarrow \text{ret}_1 \\
\overline{\text{ret_pattern}_i}^i : \text{ret}_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\
\mathcal{C}, \mathcal{C}_1; \mathcal{L}, \mathcal{L}_1; \Phi, \Phi_1; \mathcal{R}, \mathcal{R}_1 \vdash \text{expr} \Leftarrow \text{ret}_2 \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{let strong } \overline{\text{ret_pattern}_i}^i = \text{is_expr in expr} \Leftarrow \text{ret}_2
\end{array}
\quad \text{TY_IS_TE_LETS}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{expr} \Leftarrow \text{ret}}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_expr} \Leftarrow \text{ret} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_expr} \Leftarrow \text{ret}
\end{array}
\quad \text{TY_TE_IS}$$

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{seq_expr} \Leftarrow \text{ret} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{seq_expr} \Leftarrow \text{ret}
\end{array}
\quad \text{TY_TE_SEQ}$$

$$\boxed{\text{pattern} = pval \rightsquigarrow \sigma}$$

$$\frac{}{\therefore = pval \rightsquigarrow \cdot} \quad \text{SUBS_DECONS_VALUE_NO_SYM_ANNOT}$$

$$\frac{}{x:_ = pval \rightsquigarrow pval/x, \cdot} \quad \text{SUBS_DECONS_VALUE_SYM_ANNOT}$$

$$\frac{\begin{array}{l} pattern_1 = pval_1 \rightsquigarrow \sigma_1 \\ pattern_2 = pval_2 \rightsquigarrow \sigma_2 \end{array}}{\text{Cons}(pattern_1, pattern_2) = \text{Cons}(pval_1, pval_2) \rightsquigarrow \sigma_1, \sigma_2} \quad \text{SUBS_DECONS_VALUE_CONS}$$

$$\frac{\overline{pattern_i = pval_1 \rightsquigarrow \sigma_i}^i}{\text{Tuple}(\overline{pattern_i}^i) = \text{Tuple}(\overline{pval_i}^i) \rightsquigarrow \overline{\sigma_i}^i} \quad \text{SUBS_DECONS_VALUE_TUPLE}$$

$$\frac{\overline{pattern_i = pval_1 \rightsquigarrow \sigma_i}^i}{\text{Array}(\overline{pattern_i}^i) = \text{Array}(\overline{pval_i}^i) \rightsquigarrow \overline{\sigma_i}^i} \quad \text{SUBS_DECONS_VALUE_ARRAY}$$

$$\frac{pattern = pval \rightsquigarrow \sigma}{\text{Specified}(pattern) = pval \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_VALUE_SPECIFIED}$$

$$\boxed{ident_or_pattern = pval \rightsquigarrow \sigma}$$

$$\frac{}{x = pval \rightsquigarrow pval/x, \cdot} \quad \text{SUBS_DECONS_VALUE'_SYM}$$

$$\frac{pattern = pval \rightsquigarrow \sigma}{pattern = pval \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_VALUE'_PATTERN}$$

$$\boxed{res_pattern = res_term \rightsquigarrow \sigma}$$

$$\frac{}{\text{emp} = \text{emp} \rightsquigarrow \cdot} \quad \text{SUBS_DECONS_RES_EMP}$$

$$\frac{}{pt = pt \rightsquigarrow \cdot} \quad \text{SUBS_DECONS_RES_POINTS_TO}$$

$$\frac{}{ident = res_term \rightsquigarrow res_term/ident, \cdot} \quad \text{SUBS_DECONS_RES_VAR}$$

$$\frac{\begin{array}{l} res_pattern_1 = res_term_1 \rightsquigarrow \sigma_1 \\ res_pattern_2 = res_term_2 \rightsquigarrow \sigma_2 \end{array}}{\langle res_pattern_1, res_pattern_2 \rangle = \langle res_term_1, res_term_2 \rangle \rightsquigarrow \sigma_1, \sigma_2} \quad \text{SUBS_DECONS_RES_PAIR}$$

$$\frac{res_pattern = res_term \rightsquigarrow \sigma}{\mathbf{pack}(ident, res_pattern) = \mathbf{pack}(pval, res_term) \rightsquigarrow pval/ident, \sigma} \quad \text{SUBS_DECONS_RES_PACK}$$

$$\boxed{ret_pattern = spine_elem \rightsquigarrow \sigma}$$

$$\frac{ident_or_pattern = pval \rightsquigarrow \sigma}{\mathbf{comp} ident_or_pattern = pval \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_RET_COMP}$$

$$\frac{}{\mathbf{log} ident = pval \rightsquigarrow pval/ident, \cdot} \quad \text{SUBS_DECONS_RET_LOG}$$

$$\frac{res_pattern = res_term \rightsquigarrow \sigma}{\mathbf{res} res_pattern = res_term \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_RET_RES}$$

$$\boxed{\overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret}$$

$$\frac{}{:: ret \gg \cdot; ret} \quad \text{SUBS_DECONS_ARG_EMPTY}$$

$$\frac{\overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine_elem_i}^i :: \Pi x:\beta. arg \gg pval/x, \sigma; ret} \quad \text{SUBS_DECONS_ARG_COMP}$$

$$\frac{\overline{x_i = spine_elem_i^i} :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine_elem_i^i} :: \forall x:\beta. arg \gg pval/x, \sigma; ret} \quad \text{SUBS_DECONS_ARG_LOG}$$

$$\frac{\overline{x_i = spine_elem_i^i} :: arg \gg \sigma; ret}{x = res_term, \overline{x_i = spine_elem_i^i} :: res \multimap arg \gg res_term/x, \sigma; ret} \quad \text{SUBS_DECONS_ARG_RES}$$

$$\frac{\overline{x_i = spine_elem_i^i} :: arg \gg \sigma; ret}{\overline{x_i = spine_elem_i^i} :: term \supset arg \gg \sigma; ret} \quad \text{SUBS_DECONS_ARG_PHI}$$

$$\boxed{\langle pexpr \rangle \longrightarrow \langle pexpr' \rangle}$$

$$\frac{mem_ptr' \equiv mem_ptr +_{\text{ptr}} mem_int \times \text{size_of}(\tau)}{\langle \text{array_shift}(mem_ptr, \tau, mem_int) \rangle \longrightarrow \langle mem_ptr' \rangle} \quad \text{OP_PE_PE_ARRAYSHIFT}$$

$$\frac{mem_ptr' \equiv mem_ptr +_{\text{ptr}} \text{offset_of}_{tag}(member)}{\langle \text{member_shift}(mem_ptr, tag, member) \rangle \longrightarrow \langle mem_ptr' \rangle} \quad \text{OP_PE_PE_MEMBERSHIFT}$$

$$\overline{\langle \text{not}(\text{True}) \rangle} \longrightarrow \langle \text{False} \rangle \quad \text{OP_PE_PE_NOTTRUE}$$

$$\overline{\langle \text{not}(\text{False}) \rangle} \longrightarrow \langle \text{True} \rangle \quad \text{OP_PE_PE_NOTFALSE}$$

$$\frac{mem_int \equiv mem_int_1 \text{ binop}_{arith} mem_int_2}{\langle mem_int_1 \text{ binop}_{arith} mem_int_2 \rangle \longrightarrow \langle mem_int \rangle} \quad \text{OP_PE_PE_ARITH_BINOP}$$

$$\frac{bool_value \equiv mem_int_1 \text{ binop}_{rel} mem_int_2}{\langle mem_int_1 \text{ binop}_{rel} mem_int_2 \rangle \longrightarrow \langle bool_value \rangle} \quad \text{OP_PE_PE_REL_BINOP}$$

$$\frac{bool_value \equiv bool_value_1 \text{ binop}_{bool} bool_value_2}{\langle bool_value_1 \text{ binop}_{bool} bool_value_2 \rangle \longrightarrow \langle bool_value \rangle} \quad \text{OP_PE_PE_BOOL_BINOP}$$

$$\frac{}{\langle \text{assert_undef}(\text{True}, UB_name) \rangle \longrightarrow \langle \text{Unit} \rangle} \quad \text{OP_PE_PE_ASSERT_UNDEF}$$

$$\frac{}{\langle \text{bool_to_integer}(\text{True}) \rangle \longrightarrow \langle 1 \rangle} \quad \text{OP_PE_PE_BOOL_TO_INTEGER_TRUE}$$

$$\frac{}{\langle \text{bool_to_integer}(\text{False}) \rangle \longrightarrow \langle 0 \rangle} \quad \text{OP_PE_PE_BOOL_TO_INTEGER_FALSE}$$

$$\frac{\begin{array}{l} abbrev_1 \equiv \text{max_int}_\tau - \text{min_int}_\tau + 1 \\ abbrev_2 \equiv pval \text{ rem_f } abbrev_1 \\ mem_int' \equiv \text{if } abbrev_2 \leq \text{max_int}_\tau \text{ then } abbrev_2 \text{ else } abbrev_2 - abbrev_1 \end{array}}{\langle \text{wrapI}(\tau, mem_int) \rangle \longrightarrow \langle mem_int' \rangle} \quad \text{OP_PE_PE_WRAP}$$

$$\boxed{\langle pexpr \rangle \longrightarrow \langle tpepr:(y:\beta. term) \rangle}$$

$$\frac{\begin{array}{l} name: pure_arg \equiv \overline{x_i}^i \mapsto tpepr \in \text{Globals} \\ \overline{x_i = pval_i}^i :: pure_arg \gg \sigma; \Sigma y:\beta. term \wedge \text{I} \end{array}}{\langle name(\overline{pval_i}^i) \rangle \longrightarrow \langle \sigma(tpepr):(y:\beta. \sigma(term)) \rangle} \quad \text{OP_PE_TPE_CALL}$$

$$\boxed{\langle tpepr \rangle \longrightarrow \langle tpepr' \rangle}$$

$$\frac{\begin{array}{l} pattern_j = pval \rightsquigarrow \sigma_j \\ \forall i < j. \text{not } (pattern_i = pval \rightsquigarrow \sigma_i) \end{array}}{\langle \text{case } pval \text{ of } \overline{pattern_i \Rightarrow tpepr_i}^i \text{ end} \rangle \longrightarrow \langle \sigma_j(tpepr_j) \rangle} \quad \text{OP_TPE_TPE_CASE}$$

$$\frac{ident_or_pattern = pval \rightsquigarrow \sigma}{\langle \text{let } ident_or_pattern = pval \text{ in } tpepr \rangle \longrightarrow \langle \sigma(tpepr) \rangle} \quad \text{OP_TPE_TPE_LET_SUB}$$

$$\frac{\langle pexpr \rangle \longrightarrow \langle pexpr' \rangle}{\langle \text{let } ident_or_pattern = pexpr \text{ in } texpr \rangle \longrightarrow \langle \text{let } ident_or_pattern = pexpr' \text{ in } texpr \rangle} \quad \text{OP_TPE_TPE_LET_LET}$$

$$\frac{\langle pexpr \rangle \longrightarrow \langle texpr_1:(y:\beta. term) \rangle}{\langle \text{let } ident_or_pattern = pexpr \text{ in } texpr_2 \rangle \longrightarrow \langle \text{let } ident_or_pattern:(y:\beta. term) = texpr_1 \text{ in } texpr_2 \rangle} \quad \text{OP_TPE_TPE_LET_LETT}$$

$$\frac{ident_or_pattern = pval \rightsquigarrow \sigma}{\langle \text{let } ident_or_pattern:(y:\beta. term) = \text{done } pval \text{ in } texpr \rangle \longrightarrow \langle \sigma(texpr) \rangle} \quad \text{OP_TPE_TPE_LETT_SUB}$$

$$\frac{\langle texpr_1 \rangle \longrightarrow \langle texpr'_1 \rangle}{\langle \text{let } ident_or_pattern:(y:\beta. term) = texpr_1 \text{ in } texpr_2 \rangle \longrightarrow \langle \text{let } ident_or_pattern:(y:\beta. term) = texpr'_1 \text{ in } texpr_2 \rangle} \quad \text{OP_TPE_TPE_LETT_LETT}$$

$$\frac{}{\langle \text{if True then } texpr_1 \text{ else } texpr_2 \rangle \longrightarrow \langle texpr_1 \rangle} \quad \text{OP_TPE_TPE_IF_TRUE}$$

$$\frac{}{\langle \text{if False then } texpr_1 \text{ else } texpr_2 \rangle \longrightarrow \langle texpr_2 \rangle} \quad \text{OP_TPE_TPE_IF_FALSE}$$

$$\boxed{\langle seq_expr \rangle \longrightarrow \langle texpr:ret \rangle}$$

$$\frac{\begin{array}{l} pval:arg \equiv \overline{x_i}^i \mapsto texpr \in \text{Globals} \\ \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\langle \text{ccall } (\tau, pval, \overline{spine_elem_i}^i) \rangle \longrightarrow \langle \sigma(texpr):\sigma(ret) \rangle} \quad \text{OP_SE_TE_CCALL}$$

$$\frac{\begin{array}{l} name:arg \equiv \overline{x_i}^i \mapsto texpr \in \text{Globals} \\ \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret \end{array}}{\langle \text{pcall } (name, \overline{spine_elem_i}^i) \rangle \longrightarrow \langle \sigma(texpr):\sigma(ret) \rangle} \quad \text{OP_SE_TE_PCALL}$$

$$\boxed{\langle h; seq_texpr \rangle \longrightarrow \langle h'; texpr \rangle}$$

$$\frac{\text{ident:arg} \equiv \overline{x_i}^i \mapsto \text{texpr} \in \mathbf{Globals} \quad \overline{x_i = pval_i}^i :: \text{arg} \gg \sigma; \mathbf{false} \wedge \mathbf{I}}{\langle h; \mathbf{run ident} \overline{pval_i}^i \rangle \longrightarrow \langle h; \sigma(\text{texpr}) \rangle} \quad \text{OP_STE_TE_RUN}$$

$$\frac{\text{pattern}_j = pval \rightsquigarrow \sigma_j \quad \forall i < j. \mathbf{not} (\text{pattern}_i = pval \rightsquigarrow \sigma_i)}{\langle h; \mathbf{case pval of} \overline{\text{pattern}_i \Rightarrow \text{texpr}_i}^i \mathbf{end} \rangle \longrightarrow \langle h; \sigma_j(\text{texpr}_j) \rangle} \quad \text{OP_STE_TE_CASE}$$

$$\frac{\text{ident_or_pattern} = pval \rightsquigarrow \sigma}{\langle h; \mathbf{let ident_or_pattern} = pval \mathbf{in texpr} \rangle \longrightarrow \langle h; \sigma(\text{texpr}) \rangle} \quad \text{OP_STE_TE_LETP_SUB}$$

$$\frac{\langle pexpr \rangle \longrightarrow \langle pexpr' \rangle}{\langle h; \mathbf{let ident_or_pattern} = pexpr \mathbf{in texpr} \rangle \longrightarrow \langle h; \mathbf{let ident_or_pattern} = pexpr' \mathbf{in texpr} \rangle} \quad \text{OP_STE_TE_LETP_LETP}$$

$$\frac{\langle pexpr \rangle \longrightarrow \langle texpr:(y:\beta. \text{term}) \rangle}{\langle h; \mathbf{let ident_or_pattern} = pexpr \mathbf{in texpr} \rangle \longrightarrow \langle h; \mathbf{let ident_or_pattern}:(y:\beta. \text{term}) = texpr \mathbf{in texpr} \rangle} \quad \text{OP_STE_TE_LETP_LETP}$$

$$\frac{\text{ident_or_pattern} = pval \rightsquigarrow \sigma}{\langle h; \mathbf{let ident_or_pattern}:(y:\beta. \text{term}) = \mathbf{done pval} \mathbf{in texpr} \rangle \longrightarrow \langle h; \sigma(\text{texpr}) \rangle} \quad \text{OP_STE_TE_LETP_SUB}$$

$$\frac{\langle texpr \rangle \longrightarrow \langle texpr' \rangle}{\langle h; \mathbf{let ident_or_pattern}:(y:\beta. \text{term}) = texpr \mathbf{in texpr} \rangle \longrightarrow \langle h; \mathbf{let ident_or_pattern}:(y:\beta. \text{term}) = texpr' \mathbf{in texpr} \rangle} \quad \text{OP_STE_TE_LETP_LETP}$$

$$\frac{\overline{\text{ret_pattern}_i = \text{spine_elem}_i}^i \rightsquigarrow \sigma_i^i}{\langle h; \mathbf{let ret_pattern}_i^i : \mathbf{ret} = \mathbf{done spine_elem}_i^i \mathbf{in texpr} \rangle \longrightarrow \langle h; \overline{\sigma_i}^i(\text{texpr}) \rangle} \quad \text{OP_STE_TE_LETT_SUB}$$

$$\frac{\langle seq_expr \rangle \longrightarrow \langle texpr_1 : ret \rangle}{\langle h; \text{let } \overline{ret_pattern_i}^i = seq_expr \text{ in } texpr_2 \rangle \longrightarrow \langle h; \text{let } \overline{ret_pattern_i}^i : ret = texpr_1 \text{ in } texpr_2 \rangle} \quad \text{OP_STE_TE_LET_LETT}$$

$$\frac{\langle h; texpr_1 \rangle \longrightarrow \langle h'; texpr'_1 \rangle}{\langle h; \text{let } \overline{ret_pattern_i}^i : ret = texpr_1 \text{ in } texpr_2 \rangle \longrightarrow \langle h; \text{let } \overline{ret_pattern_i}^i : ret = texpr'_1 \text{ in } texpr_2 \rangle} \quad \text{OP_STE_TE_LETT_LETT}$$

$$\frac{}{\langle h; \text{if True then } texpr_1 \text{ else } texpr_2 \rangle \longrightarrow \langle h; texpr_1 \rangle} \quad \text{OP_STE_TE_IF_TRUE}$$

$$\frac{}{\langle h; \text{if False then } texpr_1 \text{ else } texpr_2 \rangle \longrightarrow \langle h; texpr_2 \rangle} \quad \text{OP_STE_TE_IF_FALSE}$$

$$\frac{}{\langle h; \text{bound } [int](is_texpr) \rangle \longrightarrow \langle h; is_texpr \rangle} \quad \text{OP_STE_TE_BOUND}$$

$$\boxed{\langle h; mem_op \rangle \longrightarrow \langle h'; tval \rangle}$$

$$\frac{mem_int \equiv \text{cast_ptr_to_int } mem_ptr}{\langle h; \text{intFromPtr } (\tau_1, \tau_2, mem_ptr) \rangle \longrightarrow \langle h; \text{done } mem_int \rangle} \quad \text{OP_MEMOP_TVAL_INTFROMPTR}$$

$$\frac{mem_ptr \equiv \text{cast_ptr_to_int } mem_int}{\langle h; \text{ptrFromInt } (\tau_1, \tau_2, mem_int) \rangle \longrightarrow \langle h; \text{done } mem_ptr \rangle} \quad \text{OP_MEMOP_TVAL_PTRFROMINT}$$

$$\frac{bool_value \equiv \text{aligned } (\tau, mem_ptr)}{\langle h + \{ mem_ptr \check{\vdash}_{\tau} - \}; \text{ptrValidForDeref } (\tau, mem_ptr, mem_ptr \check{\vdash}_{\tau} -) \rangle \longrightarrow \langle h + \{ mem_ptr \check{\vdash}_{\tau} - \}; \text{done } bool_value, mem_ptr \check{\vdash}_{\tau} - \rangle} \quad \text{OP_MEMOP_TVAL_PTRVALID}$$

$$\frac{bool_value \equiv \text{aligned } (\tau, mem_ptr)}{\langle h; \text{ptrWellAligned } (\tau, mem_ptr) \rangle \longrightarrow \langle h; \text{done } bool_value \rangle} \quad \text{OP_MEMOP_TVAL_PTRWELLALIGNED}$$

$$\frac{mem_ptr' \equiv mem_ptr +_{\text{ptr}} (mem_int \times \text{size.of}(\tau))}{\langle h; \text{ptrArrayShift}(mem_ptr, \tau, mem_int) \rangle \longrightarrow \langle h; \text{done } mem_ptr' \rangle} \quad \text{OP_MEMOP_TVAL_PTRARRAYSHIFT}$$

$$\boxed{\langle h; mem_action \rangle \longrightarrow \langle h'; tval \rangle}$$

$$\frac{\begin{array}{l} \text{fresh}(mem_ptr) \\ \text{representable}(\tau*, mem_ptr) \\ \text{alignedI}(mem_int, mem_ptr) \\ pval:\beta_\tau \end{array}}{\langle h; \text{create}(mem_int, \tau) \rangle \longrightarrow \langle h + \{mem_ptr \overset{\times}{\mapsto}_\tau pval\}; \text{done } mem_ptr, pval, mem_ptr \overset{\times}{\mapsto}_\tau pval \rangle} \quad \text{OP_ACTION_TVAL_CREATE}$$

$$\frac{}{\langle h + \{mem_ptr \overset{\checkmark}{\mapsto}_\tau pval\}; \text{load}(\tau, mem_ptr, -, mem_ptr \overset{\checkmark}{\mapsto}_\tau pval) \rangle \longrightarrow \langle h + \{mem_ptr \overset{\checkmark}{\mapsto}_\tau pval\}; \text{done } pval, mem_ptr \overset{\checkmark}{\mapsto}_\tau pval \rangle} \quad \text{OP_ACTION_TVAL_LOAD}$$

$$\frac{}{\langle h + \{mem_ptr \overset{\checkmark}{\mapsto}_\tau -\}; \text{store}(-, \tau, mem_ptr, pval, -, mem_ptr \overset{\checkmark}{\mapsto}_\tau -) \rangle \longrightarrow \langle h + \{mem_ptr \overset{\checkmark}{\mapsto}_\tau pval\}; \text{done Unit}, mem_ptr \overset{\checkmark}{\mapsto}_\tau pval \rangle} \quad \text{OP_ACTION_TVAL_STORE}$$

$$\frac{}{\langle h + \{mem_ptr \mapsto_\tau -\}; \text{kill}(\text{static } \tau, mem_ptr, mem_ptr \mapsto_\tau -) \rangle \longrightarrow \langle h; \text{done Unit} \rangle} \quad \text{OP_ACTION_TVAL_KILL_STATIC}$$

$$\boxed{\langle h; is_expr \rangle \longrightarrow \langle h'; is_expr' \rangle}$$

$$\frac{\langle h; mem_op \rangle \longrightarrow \langle h; tval \rangle}{\langle h; \text{memop}(mem_op) \rangle \longrightarrow \langle h; tval \rangle} \quad \text{OP_ISE_ISE_MEMOP}$$

$$\frac{\langle h; mem_action \rangle \longrightarrow \langle h; tval \rangle}{\langle h; mem_action \rangle \longrightarrow \langle h; tval \rangle} \quad \text{OP_ISE_ISE_ACTION}$$

$$\frac{\langle h; mem_action \rangle \longrightarrow \langle h'; tval \rangle}{\langle h; \text{neg } mem_action \rangle \longrightarrow \langle h'; tval \rangle} \quad \text{OP_ISE_ISE_NEG_ACTION}$$

$$\boxed{\langle h; is_expr \rangle \longrightarrow \langle h'; texpr \rangle}$$

$$\frac{\overline{ret_pattern_i = spine_elem_i \rightsquigarrow \sigma_i^i}}{\langle h; \text{let strong } \overline{ret_pattern_i^i = \text{done } spine_elem_i^i} \text{ in } texpr \rangle \longrightarrow \langle h; \overline{\sigma_i^i}(texpr) \rangle} \quad \text{OP_ISTE_ISTE_LETS_SUB}$$

$$\frac{\langle h; is_expr \rangle \longrightarrow \langle h'; is_expr' \rangle}{\langle h; \text{let strong } \overline{ret_pattern_i^i = is_expr} \text{ in } texpr \rangle \longrightarrow \langle h'; \text{let strong } \overline{ret_pattern_i^i = is_expr'} \text{ in } texpr \rangle} \quad \text{OP_ISTE_ISTE_LETS_LETS}$$

$$\boxed{\langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle}$$

$$\frac{\langle h; seq_texpr \rangle \longrightarrow \langle h; texpr \rangle}{\langle h; seq_texpr \rangle \longrightarrow \langle h; texpr \rangle} \quad \text{OP_TE_TE_SEQ}$$

$$\frac{\langle h; is_texpr \rangle \longrightarrow \langle h'; texpr \rangle}{\langle h; is_texpr \rangle \longrightarrow \langle h'; texpr \rangle} \quad \text{OP_TE_TE_IS}$$

$$\boxed{\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret}$$

$$\frac{}{::ret \rightsquigarrow \cdot; \cdot; \cdot \mid ret} \quad \text{ARG_ENV_RET}$$

$$\frac{\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret}{x, \overline{x_i}^i :: \Pi x:\beta. arg \rightsquigarrow \mathcal{C}, x:\beta; \mathcal{L}; \Phi; \mathcal{R} \mid ret} \quad \text{ARG_ENV_COMP}$$

$$\frac{\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret}{x, \overline{x_i}^i :: \forall x:\beta. arg \rightsquigarrow \mathcal{C}; \mathcal{L}, x:\beta; \Phi; \mathcal{R} \mid ret} \quad \text{ARG_ENV_LOG}$$

$$\frac{\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret}{\overline{x_i}^i :: term \supset arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi, term; \mathcal{R} \mid ret} \quad \text{ARG_ENV_PHI}$$

$$\frac{\overline{x_i}^i :: \text{arg} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid \text{ret}}{x, \overline{x_i}^i :: \text{res} \multimap \text{arg} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}, x:\text{res} \mid \text{ret}} \quad \text{ARG_ENV_RES}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}$$

$$\frac{}{\cdot; \cdot; \cdot; \cdot \sqsubseteq \cdot; \cdot; \cdot; \cdot} \quad \text{WEAK_EMPTY}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}, x:\beta; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}', x:\beta; \mathcal{L}'; \Phi'; \mathcal{R}'} \quad \text{WEAK_CONS_COMP}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}, x:\beta; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}', x:\beta; \Phi'; \mathcal{R}'} \quad \text{WEAK_CONS_LOG}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}; \Phi, \text{term}; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi', \text{term}; \mathcal{R}'} \quad \text{WEAK_CONS_PHI}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}, x:\text{res} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}', x:\text{res}} \quad \text{WEAK_CONS_RES}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}', x:\beta; \mathcal{L}'; \Phi'; \mathcal{R}'} \quad \text{WEAK_SKIP_COMP}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}', x:\beta; \Phi'; \mathcal{R}'} \quad \text{WEAK_SKIP_LOG}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi', \text{term}; \mathcal{R}'} \quad \text{WEAK_SKIP_PHI}$$

$$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash (\cdot):(\cdot; \cdot; \cdot; \cdot)} \text{TY_SUBS_EMPTY}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}') \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (pval/x, \sigma):(\mathcal{C}', x:\beta; \mathcal{L}'; \Phi'; \mathcal{R}')} \text{TY_SUBS_CONS_COMP}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}') \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (pval/x, \sigma):(\mathcal{C}', \mathcal{L}', x:\beta; \Phi'; \mathcal{R}')} \text{TY_SUBS_CONS_LOG}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi', term; \mathcal{R}')} \text{TY_SUBS_CONS_PHI}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}') \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res_term \Leftarrow res \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}, \mathcal{R}_1 \vdash (res_term/x, \sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}', x:res)} \text{TY_SUBS_CONS_RES}$$

Definition rules: 195 good 0 bad

Definition rule clauses: 432 good 0 bad