

<i>ident, x, y, y_p, y_f, -, abbrev, r, α</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
	$\tau*$	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> array ($\overline{loaded_value_i}^i$) (struct <i>ident</i>) { $\overline{member_i:\tau_i = mem_val_i}^i$ } (union <i>ident</i>) { <i>member</i> = <i>mem_val</i> }	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>	$::=$ specified <i>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 $\beta[\overline{value_i}^i]$ ($\overline{value_i}^i$)	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 Array Specified	data constructors empty list list cons tuple C array 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
			constrained ($\overline{mem_iv_c_i, pval_i^i}$)	constrained value
			error (<i>string</i> , <i>pval</i>)	impl-defined static error
			<i>ctor_val</i> ($\overline{pval_i^i}$)	data constructor application
			(struct <i>ident</i>){ $\overline{.member_i = pval_i^i}$ }	C struct expression
			(union <i>ident</i>){ $\overline{.member = pval}$ }	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
			$_:\beta$	binders = {}

		$ident:\beta$	$binders = ident$	
$pattern$	$::=$			
		$ident_opt_ \beta$	$binders = binders(ident_opt_ \beta)$	
		$ctor_val(\overline{pattern_i}^i)$	$binders = binders(\overline{pattern_i}^i)$	
z	$::=$			OCaml arbitrary-width integer
		i	M	literal integer
		mem_int	M	
		$size_of(\tau)$	M	size of a C type
		$offset_of_{tag}(member)$	M	offset of a struct member
		ptr_size	M	size of a pointer
		max_int_τ	M	maximum value of int of type τ
		min_int_τ	M	minimum value of int of type τ
$\mathbb{Q}, q, -$	$::=$			OCaml type for rational numbers
		$\frac{int_1}{int_2}$		
lit	$::=$			
		$ident$		
		unit		
		$bool$		
		z		
		\mathbb{Q}		
$ident_or_pattern$	$::=$			
		$ident$	$binders = ident$	
		$pattern$	$binders = binders(pattern)$	
$bool_op$	$::=$			
		$\neg term$		

		$term_1 = term_2$	
		$term_1 \rightarrow term_2$	
		$\bigwedge(\overline{term_i}^i)$	
		$\bigvee(\overline{term_i}^i)$	
		$term_1 \text{ binop}_{bool} term_2$	M
		if $term_1$ then $term_2$ else $term_3$	
$arith_op$	$::=$		
		$term_1 + term_2$	
		$term_1 - term_2$	
		$term_1 \times term_2$	
		$term_1 / term_2$	
		$term_1 \text{ rem_t } term_2$	
		$term_1 \text{ rem_f } term_2$	
		$term_1 \wedge term_2$	
		$term_1 \text{ binop}_{arith} term_2$	M
cmp_op	$::=$		
		$term_1 < term_2$	less than
		$term_1 \leq term_2$	less than or equal
		$term_1 \text{ binop}_{rel} term_2$	M
$list_op$	$::=$		
		nil	
		$term_1 :: term_2$	
		tl $term$	
		$term^{(int)}$	
$tuple_op$	$::=$		
		$(\overline{term_i}^i)$	
		$term^{(int)}$	

<i>pointer_op</i>	::=	<ul style="list-style-type: none"> <i>mem_ptr</i> <i>term</i>₁ +_{ptr} <i>term</i>₂ cast_int_to_ptr <i>term</i> cast_ptr_to_int <i>term</i>
<i>array_op</i>	::=	<ul style="list-style-type: none"> $[\overline{term_i}^i]$ <i>term</i>₁[<i>term</i>₂]
<i>param_op</i>	::=	<ul style="list-style-type: none"> <i>ident</i>:β. <i>term</i> <i>term</i>(<i>term</i>₁, .., <i>term</i>_{<i>n</i>})
<i>struct_op</i>	::=	<ul style="list-style-type: none"> <i>term.member</i>
<i>ct_pred</i>	::=	<ul style="list-style-type: none"> representable (τ, <i>term</i>) aligned (τ, <i>term</i>) alignedI (<i>term</i>₁, <i>term</i>₂)
<i>term</i> , -	::=	<ul style="list-style-type: none"> <i>lit</i> <i>arith_op</i> <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
		$\sigma(term)$	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 access
		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>tpexpr</i>	bind binders(<i>ident_or_pattern</i>) in <i>tpexpr</i>		pure let
		let <i>ident_or_pattern</i> :($y_1:\beta_1. term_1$) = <i>tpexpr</i> ₁ in <i>tpexpr</i> ₂	bind binders(<i>ident_or_pattern</i>) in <i>tpexpr</i> ₂		annotated pure let
			bind y_1 in <i>term</i> ₁		
		if <i>pval</i> then <i>tpexpr</i> ₁ else <i>tpexpr</i> ₂			pure if
		$\sigma(tpexpr)$	M		simul-sub σ in <i>tpexpr</i>
<i>tpexpr_case_branch</i>	::=				pure top-level case expression

	$pattern \Rightarrow tpepr$	bind binders($pattern$) in $tpepr$	top-level case expression branch
m_kill_kind	$::=$ dynamic static τ		
$bool, _$	$::=$ true false		OCaml booleans
$int, _$	$::=$ i		OCaml fixed-width integer literal integer
res_term	$::=$ emp $points_to$ $ident$ $\langle res_term_1, res_term_2 \rangle$ pack ($pval, res_term$) fold (res_term) $\sigma(res_term)$	M	resource terms empty heap single-cell heap variable seperating-conjunction pair packing for existentials fold into recursive res. pred. substitution for resource terms
mem_action	$::=$ create ($pval, \tau$) create_readonly ($pval_1, \tau, pval_2$) alloc ($pval_1, pval_2$) kill ($m_kill_kind, pval, pt$) store ($bool, \tau, pval_1, pval_2, mem_order, pt$) load ($\tau, pval, mem_order, pt$) rmw ($\tau, pval_1, pval_2, pval_3, mem_order_1, mem_order_2$) fence (mem_order)		memory actions true means store is locking

	\mid <code>cmp_exch_strong</code> ($\tau, pval_1, pval_2, pval_3, mem_order_1, mem_order_2$) \mid <code>cmp_exch_weak</code> ($\tau, pval_1, pval_2, pval_3, mem_order_1, mem_order_2$) \mid <code>linux_fence</code> (<code>linux_mem_order</code>) \mid <code>linux_load</code> ($\tau, pval, linux_mem_order$) \mid <code>linux_store</code> ($\tau, pval_1, pval_2, linux_mem_order$) \mid <code>linux_rmw</code> ($\tau, pval_1, pval_2, linux_mem_order$)	
<i>polarity</i>	$::=$ \mid \mid <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>	$::=$ \mid <code>polarity mem_action</code>	memory actions with polarity
<i>mem_op</i>	$::=$ \mid <code>pval₁ binop_{rel} pval₂</code> \mid <code>pval₁ −_τ pval₂</code> \mid <code>intFromPtr</code> ($\tau_1, \tau_2, pval$) \mid <code>ptrFromInt</code> ($\tau_1, \tau_2, pval$) \mid <code>ptrValidForDeref</code> ($\tau, pval, pt$) \mid <code>ptrWellAligned</code> ($\tau, pval$) \mid <code>ptrArrayShift</code> (<code>pval₁</code> , τ , <code>pval₂</code>) \mid <code>memcpy</code> (<code>pval₁</code> , <code>pval₂</code> , <code>pval₃</code>) \mid <code>memcmp</code> (<code>pval₁</code> , <code>pval₂</code> , <code>pval₃</code>) \mid <code>realloc</code> (<code>pval₁</code> , <code>pval₂</code> , <code>pval₃</code>) \mid <code>va_start</code> (<code>pval₁</code> , <code>pval₂</code>) \mid <code>va_copy</code> (<code>pval</code>) \mid <code>va_arg</code> (<code>pval</code> , τ) \mid <code>va_end</code> (<code>pval</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
<i>spine_elem</i>	$::=$	spine element

		$pval$		pure or logical value
		res_term		resource value
		$\sigma(spine_elem)$	M	substitution for spine elements / return values
$spine$	$::=$	$\overline{spine_elem_i}^i$		spine
$tval$	$::=$	done $spine$		(effectful) top-level values
		undef UB_name		end of top-level expression
				undefined behaviour
$res_pattern$	$::=$	emp	binders = {}	resource terms
		$ident$	binders = $ident$	empty heap
		fold ($res_pattern$)	binders = {}	variable
		$\langle res_pattern_1, res_pattern_2 \rangle$	binders = binders($res_pattern_1$) \cup binders($res_pattern_2$)	unfold (recursive) predicate
		pack ($ident, res_pattern$)	binders = $ident \cup$ binders($res_pattern$)	seperating-conjunction pair
				packing for existentials
$ret_pattern$	$::=$	comp $ident_or_pattern$	binders = binders($ident_or_pattern$)	return pattern
		log $ident$	binders = $ident$	computational variable
		res $res_pattern$	binders = binders($res_pattern$)	logical variable
				resource variable
$init,$	$::=$	\checkmark		initialisation status
		\times		initialised
				uninitilised
$points_to, pt$	$::=$	$term_1 \xrightarrow{init}_\tau term_2$		points-to separation logic predicate
res	$::=$			resources

	emp points_to $\text{res}_1 * \text{res}_2$ $\exists \text{ident}:\beta. \text{res}$ $\text{term} \wedge \text{res}$ $\text{if term then res}_1 \text{ else res}_2$ $\alpha(\overline{\text{pval}_i}^i)$ $\sigma(\text{res})$	M	empty heap points-top heap pred. seperating conjunction existential logical conjunction ordered disjuction predicate simul-sub σ in res
$\text{ret}, -$	$::=$ $\Sigma \text{ident}:\beta. \text{ret}$ $\exists \text{ident}:\beta. \text{ret}$ $\text{res} \otimes \text{ret}$ $\text{term} \wedge \text{ret}$ \mathbf{I} $\sigma(\text{ret})$	M	return types return a computational value return a logical value return a resource value return a predicate (post-condition) end return list simul-sub σ in ret
seq_expr	$::=$ $\text{ccall}(\tau, \text{ident}, \text{spine})$ $\text{pcall}(\text{name}, \text{spine})$		sequential (effectful) expressions C function call procedure call
seq_texpr	$::=$ $tval$ $\text{run ident } \overline{\text{pval}_i}^i$ $\text{let ident_or_pattern} = \text{pexpr in texpr}$ $\text{let ident_or_pattern}:(y_1:\beta_1. \text{term}_1) = \text{tpexpr in texpr}$ $\text{let } \overline{\text{ret_pattern}_i}^i = \text{seq_expr in texpr}$ $\text{let } \overline{\text{ret_pattern}_i}^i:\text{ret} = \text{texpr}_1 \text{ in texpr}_2$ $\text{case pval of } \mid \text{texpr_case_branch}_i^i \text{ end}$ $\text{if pval then texpr}_1 \text{ else texpr}_2$	bind binders(ident_or_pattern) in texpr bind binders(ident_or_pattern) in texpr bind y_1 in term_1 bind binders($\overline{\text{ret_pattern}_i}^i$) in texpr bind binders($\overline{\text{ret_pattern}_i}^i$) in texpr_2	sequential top-level (effectful) expressions (effectful) top-level values run from label pure let annotated pure let bind return patterns annotated bind return patterns pattern matching conditional

		$\text{bound}[int](is_texpr)$		limit scope of indet seq behaviour, absent at ru
$texpr_case_branch$	$::=$	$pattern \Rightarrow texpr$	bind binders($pattern$) in $texpr$	top-level case expression branch top-level case expression branch
is_expr	$::=$	$tval$ $\text{memop}(mem_op)$ pol_mem_action		indet seq (effectful) expressions (effectful) top-level values pointer op involving memory memory action
is_texpr	$::=$	$\text{let weak } \overline{ret_pattern_i}^i = is_expr \text{ in } texpr$ $\text{let strong } \overline{ret_pattern_i}^i = is_expr \text{ in } texpr$	bind binders($\overline{ret_pattern_i}^i$) in $texpr$ bind binders($\overline{ret_pattern_i}^i$) in $texpr$	indet seq top-level (effectful) expressions weak sequencing strong sequencing
$texpr$	$::=$	seq_texpr is_texpr $\sigma(texpr)$	M	top-level (effectful) expressions sequential (effectful) expressions indet seq (effectful) expressions simul-sub σ in $texpr$
arg	$::=$	$\Pi ident:\beta. arg$ $\forall ident:\beta. arg$ $res \multimap arg$ $term \supset arg$ ret $\sigma(arg)$	M	argument/function types simul-sub σ in arg
$pure_arg$	$::=$	$\Pi ident:\beta. pure_arg$ $term \supset pure_arg$ $pure_ret$		pure argument/function types

$pure_ret$	$::=$ $ $ $\Sigma ident:\beta. pure_ret$ $ $ $term \wedge pure_ret$ $ $ I	pure return types
\mathcal{C}	$::=$ $ $ $.$ $ $ $\mathcal{C}, ident:\beta$ $ $ $\overline{\mathcal{C}_i}^i$	computational var env
\mathcal{L}	$::=$ $ $ $.$ $ $ $\overline{\mathcal{L}_i}^i$ $ $ $\mathcal{L}, ident:\beta$	logical var env
Φ	$::=$ $ $ $.$ $ $ $\Phi, term$ $ $ $\overline{\Phi_i}^i$	constraints env
\mathcal{R}	$::=$ $ $ $.$ $ $ $\mathcal{R}, ident:res$ $ $ $\overline{\mathcal{R}_i}^i$	resources env
σ, ψ	$::=$ $ $ $.$ $ $ $spine_elem/ident, \sigma$ $ $ $term/ident, \sigma$ $ $ $\overline{\sigma_i}^i$ $ $ $\sigma(\psi)$	substitutions M apply σ to all elements in ψ

<i>typing</i>	$ \begin{array}{l} ::= \\ \text{ \texttt{smt} } (\Phi \Rightarrow \textit{term}) \\ \textit{ident} : \beta \in \mathcal{C} \\ \textit{ident} : \beta \in \mathcal{L} \\ \text{ \texttt{struct tag} } \& \overline{\textit{member}_i : \tau_i}^i \in \text{Globals} \\ \alpha \equiv \overline{x_i : \beta_i}^i \mapsto \textit{res} \in \text{Globals} \\ \frac{}{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \textit{mem_val}_i \Rightarrow \text{mem } \overline{\beta_i}^i} \end{array} $	<p>recursive resource predicate dependent on memory object model</p>
<i>opsem</i>	$ \begin{array}{l} ::= \\ \forall i < j. \text{ \texttt{not} } (\textit{pattern}_i = \textit{pval} \rightsquigarrow \sigma_i) \\ \text{ \texttt{fresh} } (\textit{mem_ptr}) \\ \textit{term} \\ \textit{pval} : \beta \end{array} $	
<i>formula</i>	$ \begin{array}{l} ::= \\ \textit{judgement} \\ \textit{typing} \\ \textit{opsem} \\ \textit{term} \equiv \textit{term}' \\ \textit{name} : \textit{pure_arg} \equiv \overline{x_i}^i \mapsto \textit{texpr} \in \text{Globals} \\ \textit{name} : \textit{arg} \equiv \overline{x_i}^i \mapsto \textit{texpr} \in \text{Globals} \end{array} $	
<i>heap, h</i>	$ \begin{array}{l} ::= \\ \cdot \\ h + \{\textit{points_to}\} \end{array} $	heaps
<i>lemma_jtype</i>	$ \begin{array}{l} ::= \\ \overline{x_i}^i :: \textit{arg} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid \textit{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}') \end{array} $	

res_jtype	$::=$ $ \quad \Phi \vdash res \equiv res'$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow res$ $ \quad h:\mathcal{R}$
$object_value_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi \vdash object_value \Rightarrow \mathbf{obj} \beta$
$pval_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta$
$spine_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret$
$pexpr_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow ident:\beta. term$
$comp_pattern_jtype$	$::=$ $ \quad pattern:\beta \rightsquigarrow \mathcal{C} \mathbf{with} term$ $ \quad ident_or_pattern:\beta \rightsquigarrow \mathcal{C} \mathbf{with} term$
$res_pattern_jtype$	$::=$ $ \quad \Phi \vdash res' = \mathbf{strip_ifs}(res)$ $ \quad \Phi \vdash res \mathbf{as} res_pattern \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'$ $ \quad \Phi \vdash res_pattern:res \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'$
$ret_pattern_jtype$	$::=$ $ \quad \Phi \vdash \overline{ret_pattern_i}^i : ret \rightsquigarrow \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'$
$tpval_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi \vdash tpval \Leftarrow ident:\beta. term$

$tpexpr_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr \Leftarrow ident:\beta.term$
$action_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_action \Rightarrow ret$
$memop_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem_op \Rightarrow ret$
$tval_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret$
seq_expr_jtype	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_expr \Rightarrow ret$
is_expr_jtype	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is_expr \Rightarrow ret$
$texpr_jtype$	$::=$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq_texpr \Leftarrow ret$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is_texpr \Leftarrow ret$ $ \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash texpr \Leftarrow ret$
$subs_jtype$	$::=$ $ \quad pattern = pval \rightsquigarrow \sigma$ $ \quad ident_or_pattern = pval \rightsquigarrow \sigma$ $ \quad res_pattern = res_term \rightsquigarrow \sigma$ $ \quad \frac{ret_pattern_i = spine_elem_i}{x_i = spine_elem_i}^i \rightsquigarrow \sigma$ $ \quad \frac{}{x_i = spine_elem_i}^i :: arg \gg \sigma; ret$
$pure_opsem_jtype$	$::=$

$$\begin{array}{l}
| \quad \langle pexpr \rangle \longrightarrow \langle pexpr' \rangle \\
| \quad \langle pexpr \rangle \longrightarrow \langle tpepr:(y:\beta. term) \rangle \\
| \quad \langle tpepr \rangle \longrightarrow \langle tpepr' \rangle
\end{array}$$

$$\begin{array}{lcl}
opsem_jtype & ::= & \\
| & \langle h; seq_expr \rangle \longrightarrow \langle h'; texpr:ret \rangle \\
| & \langle h; seq_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
| & \langle h; mem_op \rangle \longrightarrow \langle h'; tval \rangle \\
| & \langle h; mem_action \rangle \longrightarrow \langle h'; tval \rangle \\
| & \langle h; is_expr \rangle \longrightarrow \langle h'; is_expr' \rangle \\
| & \langle h; is_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
| & \langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle
\end{array}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}') \\ \text{smt}(\Phi \Rightarrow term) \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi', term; \mathcal{R}')} \quad \text{TY_SUBS_CONS_PHI}$$

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

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

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

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

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

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

$$\frac{\text{smt}(\Phi \Rightarrow (term \rightarrow term') \wedge (term' \rightarrow term)) \quad \Phi \vdash res \equiv res'}{\Phi \vdash term \wedge res \equiv term' \wedge res'} \quad \text{TY_RES_EQ_TERM}$$

$$\frac{\text{smt}(\Phi \Rightarrow (term_1 \rightarrow term_2) \wedge (term_2 \rightarrow term_1)) \quad \Phi \vdash res_{11} \equiv res_{21} \quad \Phi \vdash res_{21} \equiv res_{22}}{\Phi \vdash \text{if } term_1 \text{ then } res_{11} \text{ else } res_{12} \equiv \text{if } term_2 \text{ then } res_{21} \text{ else } res_{22}} \quad \text{TY_RES_EQ_ORDDISJ}$$

$$\frac{}{\Phi \vdash \alpha(\overline{pval_i^i}) \equiv \alpha(\overline{pval_i^i})} \quad \text{TY_RES_EQ_PRED}$$

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

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

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

$$\frac{\Phi \vdash res'_1 = \text{strip_ifs}(res_1) \quad \Phi \vdash res'_2 = \text{strip_ifs}(res_2) \quad \Phi \vdash res'_1 \equiv res'_2}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, r : res_1 \vdash r \Leftarrow res_2} \quad \text{TY_RES_VAR}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res_term_1 \Leftarrow res_1 \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash res_term_2 \Leftarrow res_2}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \langle res_term_1, res_term_2 \rangle \Leftarrow res_1 * res_2} \quad \text{TY_RES_SEPCONJ}$$

$$\frac{\text{smt}(\Phi \Rightarrow \text{term}) \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{res_term} \Leftarrow \text{res}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{res_term} \Leftarrow \text{term} \wedge \text{res}} \quad \text{TY_RES_CONJ}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{pval} \Rightarrow \beta \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{res_term} \Leftarrow \text{pval}/y, \cdot(\text{res})}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{pack}(\text{pval}, \text{res_term}) \Leftarrow \exists y:\beta. \text{res}} \quad \text{TY_RES_PACK}$$

$$\frac{\alpha \equiv \overline{x_i:\beta_i}^i \mapsto \text{res} \in \text{Globals} \quad \mathcal{C}; \mathcal{L}; \Phi \vdash \text{pval}_i \Rightarrow \beta_i^i \quad \Phi \vdash \text{res}' = \text{strip_ifs}(\overline{\text{pval}_i/x_i, \cdot}^i(\text{res})) \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{res_term} \Leftarrow \text{res}'}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{fold}(\text{res_term}) \Leftarrow \alpha(\overline{\text{pval}_i}^i)} \quad \text{TY_RES_FOLD}$$

$$\boxed{h:\mathcal{R}}$$

$$\frac{}{\therefore} \quad \text{TY_HEAP_EMP}$$

$$\frac{h:\mathcal{R} \quad \therefore; \therefore; \mathcal{R}' \vdash pt \Leftarrow pt}{h + \{pt\}:\mathcal{R}, \mathcal{R}'} \quad \text{TY_HEAP_POINTSTO}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boxed{\text{pattern}:\beta \rightsquigarrow \mathcal{C} \text{ with } term}$

$$\frac{}{.: \beta : \beta \rightsquigarrow \cdot \text{with } _} \quad \text{TY_PAT_COMP_NO_SYM_ANNOT}$$

$$\frac{}{x:\beta:\beta \rightsquigarrow \cdot, x:\beta \text{ with } x} \quad \text{TY_PAT_COMP_SYM_ANNOT}$$

$$\frac{}{\text{Nil } \beta():\text{list } \beta \rightsquigarrow \cdot \text{with nil}} \quad \text{TY_PAT_COMP_NIL}$$

$$\frac{\begin{array}{c} pattern_1:\beta \rightsquigarrow \mathcal{C}_1 \text{ with } term_1 \\ pattern_2:\text{list } \beta \rightsquigarrow \mathcal{C}_2 \text{ with } term_2 \end{array}}{\text{Cons}(pattern_1, pattern_2):\text{list } \beta \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2 \text{ with } term_1 :: term_2} \quad \text{TY_PAT_COMP_CONS}$$

$$\frac{\overline{pattern_i:\beta_i \rightsquigarrow \mathcal{C}_i \text{ with } term_i}^i}{\text{Tuple}(\overline{pattern_i}^i):\overline{\beta_i}^i \rightsquigarrow \overline{\mathcal{C}_i}^i \text{ with } (\overline{term_i}^i)} \quad \text{TY_PAT_COMP_TUPLE}$$

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

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

$\boxed{\text{ident_or_pattern}:\beta \rightsquigarrow \mathcal{C} \text{ with } term}$

$$\frac{}{x:\beta \rightsquigarrow \cdot, x:\beta \text{ with } x} \quad \text{TY_PAT_SYM_OR_PATTERN_SYM}$$

$$\frac{\text{pattern}:\beta \rightsquigarrow \mathcal{C} \text{ with term}}{\text{pattern}:\beta \rightsquigarrow \mathcal{C} \text{ with term}} \quad \text{TY_PAT_SYM_OR_PATTERN_PATTERN}$$

$$\boxed{\Phi \vdash \text{res}' = \text{strip_ifs}(\text{res})}$$

$$\frac{}{\Phi \vdash \text{emp} = \text{strip_ifs}(\text{emp})} \quad \text{TY_PAT_RES_STRIP_IFS_EMPTY}$$

$$\frac{}{\Phi \vdash pt = \text{strip_ifs}(pt)} \quad \text{TY_PAT_RES_STRIP_IFS_POINTS_TO}$$

$$\frac{}{\Phi \vdash \text{res}_1 * \text{res}_2 = \text{strip_ifs}(\text{res}_1 * \text{res}_2)} \quad \text{TY_PAT_RES_STRIP_IFS_SEP_CONJ}$$

$$\frac{}{\Phi \vdash \exists x:\beta. \text{res} = \text{strip_ifs}(\exists x:\beta. \text{res})} \quad \text{TY_PAT_RES_STRIP_IFS_EXISTS}$$

$$\frac{}{\Phi \vdash \text{term} \wedge \text{res} = \text{strip_ifs}(\text{term} \wedge \text{res})} \quad \text{TY_PAT_RES_STRIP_IFS_TERM_CONJ}$$

$$\frac{\begin{array}{l} \text{smt}(\Phi \Rightarrow \text{term}) \\ \Phi \vdash \text{res}'_1 = \text{strip_ifs}(\text{res}'_1) \end{array}}{\Phi \vdash \text{res}'_1 = \text{strip_ifs}(\text{if } \text{term} \text{ then } \text{res}_1 \text{ else } \text{res}_2)} \quad \text{TY_PAT_RES_STRIP_IFS_TRUE}$$

$$\frac{\begin{array}{l} \text{smt}(\Phi \Rightarrow \neg \text{term}) \\ \Phi \vdash \text{res}'_2 = \text{strip_ifs}(\text{res}_2) \end{array}}{\Phi \vdash \text{res}'_2 = \text{strip_ifs}(\text{if } \text{term} \text{ then } \text{res}_1 \text{ else } \text{res}_2)} \quad \text{TY_PAT_RES_STRIP_IFS_FALSE}$$

$$\frac{}{\Phi \vdash \text{if } term \text{ then } res_1 \text{ else } res_2 = \text{strip_ifs}(\text{if } term \text{ then } res_1 \text{ else } res_2)} \quad \text{TY_PAT_RES_STRIP_IFS_UNDERDET}$$

$$\frac{}{\Phi \vdash \alpha(\overline{pval_i}^i) = \text{strip_ifs}(\alpha(\overline{pval_i}^i))} \quad \text{TY_PAT_RES_STRIP_IFS_PRED}$$

$$\boxed{\Phi \vdash res \text{ as } res_pattern \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}$$

$$\frac{}{\Phi \vdash \text{emp as emp} \rightsquigarrow \cdot; \cdot; \cdot} \quad \text{TY_PAT_RES_MATCH_EMPTY}$$

$$\frac{}{\Phi \vdash res \text{ as } r \rightsquigarrow \cdot; \cdot; \cdot, r:res} \quad \text{TY_PAT_RES_MATCH_VAR}$$

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

$$\frac{\Phi \vdash res_pattern:res \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}{\Phi \vdash term \wedge res \text{ as } res_pattern \rightsquigarrow \mathcal{L}'; \Phi', term; \mathcal{R}'} \quad \text{TY_PAT_RES_MATCH_CONJ}$$

$$\frac{\Phi \vdash res_pattern:x/y, \cdot(res) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}{\Phi \vdash \exists y:\beta. res \text{ as pack}(x, res_pattern) \rightsquigarrow \mathcal{L}', x:\beta; \Phi'; \mathcal{R}'} \quad \text{TY_PAT_RES_MATCH_PACK}$$

$$\frac{\begin{array}{l} \alpha \equiv \overline{x_i:\beta_i}^i \mapsto res \in \text{Globals} \\ \Phi \vdash res_pattern:\overline{pval_i/x_i, \cdot}^i(res) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}' \end{array}}{\Phi \vdash \alpha(\overline{pval_i}^i) \text{ as fold}(res_pattern) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'} \quad \text{TY_PAT_RES_MATCH_FOLD}$$

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

$$\frac{\begin{array}{l} \Phi \vdash res' = \mathbf{strip_ifs}(res) \\ \Phi \vdash res' \mathbf{as} res_pattern \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}' \end{array}}{\Phi \vdash res_pattern:res \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'} \quad \text{TY_PAT_RES_STRIP_IFS}$$

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

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

$$\frac{\begin{array}{l} ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1 \mathbf{with} term_1 \\ \Phi \vdash \overline{ret_pattern_i}^i : term_1/y, \cdot (ret) \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\Phi \vdash \mathbf{comp} ident_or_pattern, \overline{ret_pattern_i}^i : \Sigma y:\beta. ret \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2} \quad \text{TY_PAT_RET_COMP}$$

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

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

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

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

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

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

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool} \quad \mathcal{C}; \mathcal{L}; \Phi, pval = \text{true} \vdash texpr_1 \Leftarrow y:\beta. term \quad \mathcal{C}; \mathcal{L}; \Phi, pval = \text{false} \vdash texpr_2 \Leftarrow y:\beta. term}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{if } pval \text{ then } texpr_1 \text{ else } texpr_2 \Leftarrow y:\beta. term} \quad \text{TY_TPE_IF}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y_1:\beta_1. term_1 \quad ident_or_pattern:\beta_1 \rightsquigarrow \mathcal{C}_1 \text{ with } term \quad \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term/y_1, \cdot(term_1) \vdash texpr \Leftarrow y_2:\beta_2. term_2}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{let } ident_or_pattern = pexpr \text{ in } texpr \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_LET}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash texpr_1 \Leftarrow y_1:\beta_1. term_1 \quad ident_or_pattern:\beta_1 \rightsquigarrow \mathcal{C}_1 \text{ with } term \quad \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term/y_1, \cdot(term_1) \vdash texpr \Leftarrow y_2:\beta_2. term_2}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{let } ident_or_pattern:(y_1:\beta_1. term_1) = texpr_1 \text{ in } texpr_2 \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_LETT}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta_1 \quad \overline{pattern_i:\beta_1 \rightsquigarrow \mathcal{C}_i \text{ with } term_i}^i \quad \overline{\mathcal{C}, \mathcal{C}_i; \mathcal{L}; \Phi, term_i = pval \vdash texpr_i \Leftarrow y_2:\beta_2. term_2}^i}{\mathcal{C}; \mathcal{L}; \Phi \vdash \text{case } pval \text{ of } \mid pattern_i \Rightarrow texpr_i \text{ end} \Leftarrow y_2:\beta_2. term_2} \quad \text{TY_TPE_CASE}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y:\beta. term \\ ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1 \text{ with } term_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term_1/y, \cdot(term); \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{let } ident_or_pattern = pexpr \text{ in } texpr \Leftarrow ret} \text{TY_SEQ_TE_LETP}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash texpr \Leftarrow y:\beta. term \\ ident_or_pattern:\beta \rightsquigarrow \mathcal{C}_1 \text{ with } term_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term_1/y, \cdot(term); \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{let } ident_or_pattern:(y:\beta. term) = texpr \text{ in } texpr \Leftarrow ret} \text{TY_SEQ_TE_LETPPT}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l} \text{res_pattern}_1 = \text{res_term}_1 \rightsquigarrow \sigma_1 \\ \text{res_pattern}_2 = \text{res_term}_2 \rightsquigarrow \sigma_2 \end{array}}{\langle \text{res_pattern}_1, \text{res_pattern}_2 \rangle = \langle \text{res_term}_1, \text{res_term}_2 \rangle \rightsquigarrow \sigma_1, \sigma_2} \quad \text{SUBS_DECONS_RES_PAIR}$$

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

$$\frac{\text{res_pattern} = \text{res_term} \rightsquigarrow \sigma}{\text{fold}(\text{res_pattern}) = \text{res_term} \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_RES_FOLD}$$

$$\boxed{\overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \sigma}$$

$$\frac{}{\rightsquigarrow \cdot} \quad \text{SUBS_DECONS_RET_EMPTY}$$

$$\frac{\begin{array}{l} \text{ident_or_pattern} = \text{pval} \rightsquigarrow \sigma \\ \overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \psi \end{array}}{\text{comp } \text{ident_or_pattern} = \text{pval}, \overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \sigma, \psi} \quad \text{SUBS_DECONS_RET_COMP}$$

$$\frac{\overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \psi}{\text{log } \text{ident} = \text{pval}, \overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \text{pval}/\text{ident}, \psi} \quad \text{SUBS_DECONS_RET_LOG}$$

$$\frac{\begin{array}{l} \text{res_pattern} = \text{res_term} \rightsquigarrow \sigma \\ \overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \psi \end{array}}{\text{res } \text{res_pattern} = \text{res_term}, \overline{\text{ret_pattern}_i = \text{spine_elem}_i^i} \rightsquigarrow \sigma, \psi} \quad \text{SUBS_DECONS_RET_RES}$$

$$\boxed{x_i = \text{spine_elem}_i^i :: \text{arg} \gg \sigma; \text{ret}}$$

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

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

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

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

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

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

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

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

$$\frac{}{\langle \text{not}(\text{True}) \rangle \longrightarrow \langle \text{False} \rangle} \quad \text{OP_PE_PE_NOT_TRUE}$$

$$\frac{}{\langle \text{not}(\text{False}) \rangle \longrightarrow \langle \text{True} \rangle} \quad \text{OP_PE_PE_NOT_FALSE}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\langle pexpr \rangle \longrightarrow \langle tpepr_1:(y:\beta. term) \rangle}{\langle \text{let } ident_or_pattern = pexpr \text{ in } tpepr_2 \rangle \longrightarrow \langle \text{let } ident_or_pattern:(y:\beta. term) = tpepr_1 \text{ in } tpepr_2 \rangle} \quad \text{OP_TPE_TPE_LET_LETT}$$

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

$$\frac{\langle tpepr_1 \rangle \longrightarrow \langle tpepr'_1 \rangle}{\langle \text{let } ident_or_pattern:(y:\beta. term) = tpepr_1 \text{ in } tpepr_2 \rangle \longrightarrow \langle \text{let } ident_or_pattern:(y:\beta. term) = tpepr'_1 \text{ in } tpepr_2 \rangle} \quad \text{OP_TPE_TPE_LETT_LETT}$$

$$\frac{}{\langle \text{if True then } tpepr_1 \text{ else } tpepr_2 \rangle \longrightarrow \langle tpepr_1 \rangle} \quad \text{OP_TPE_TPE_IF_TRUE}$$

$$\frac{}{\langle \text{if False then } tpepr_1 \text{ else } tpepr_2 \rangle \longrightarrow \langle tpepr_2 \rangle} \quad \text{OP_TPE_TPE_IF_FALSE}$$

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

$$\frac{\frac{\text{ident}:\text{arg} \equiv \overline{x_i}^i \mapsto \text{texpr} \in \text{Globals}}{\overline{x_i = \text{spine_elem}_i}^i :: \text{arg} \gg \sigma; \text{ret}}}{\langle h; \text{ccall}(\tau, \text{ident}, \overline{\text{spine_elem}_i}^i) \rangle \longrightarrow \langle h; \sigma(\text{texpr}); \sigma(\text{ret}) \rangle} \text{OP_SE_TE_CCALL}$$

$$\frac{\frac{\text{name}:\text{arg} \equiv \overline{x_i}^i \mapsto \text{texpr} \in \text{Globals}}{\overline{x_i = \text{spine_elem}_i}^i :: \text{arg} \gg \sigma; \text{ret}}}{\langle h; \text{pcall}(\text{name}, \overline{\text{spine_elem}_i}^i) \rangle \longrightarrow \langle h; \sigma(\text{texpr}); \sigma(\text{ret}) \rangle} \text{OP_SE_TE_PCALL}$$

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

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

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

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

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

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

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

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

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

$$\frac{\langle h; \text{seq_expr} \rangle \longrightarrow \langle h; \text{texpr}_1 : \text{ret} \rangle}{\langle h; \text{let } \overline{\text{ret_pattern}_i^i} = \text{seq_expr in } \text{texpr}_2 \rangle \longrightarrow \langle h; \text{let } \overline{\text{ret_pattern}_i^i} : \text{ret} = \text{texpr}_1 \text{ in } \text{texpr}_2 \rangle} \quad \text{OP_STE_TE_LET_LETT}$$

$$\frac{\langle h; \text{texpr}_1 \rangle \longrightarrow \langle h'; \text{texpr}'_1 \rangle}{\langle h; \text{let } \overline{\text{ret_pattern}_i^i} : \text{ret} = \text{texpr}_1 \text{ in } \text{texpr}_2 \rangle \longrightarrow \langle h'; \text{let } \overline{\text{ret_pattern}_i^i} : \text{ret} = \text{texpr}'_1 \text{ in } \text{texpr}_2 \rangle} \quad \text{OP_STE_TE_LETT_LETT}$$

$$\frac{}{\langle h; \text{if True then } \text{texpr}_1 \text{ else } \text{texpr}_2 \rangle \longrightarrow \langle h; \text{texpr}_1 \rangle} \quad \text{OP_STE_TE_IF_TRUE}$$

$$\frac{}{\langle h; \text{if False then } \text{texpr}_1 \text{ else } \text{texpr}_2 \rangle \longrightarrow \langle h; \text{texpr}_2 \rangle} \quad \text{OP_STE_TE_IF_FALSE}$$

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

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

$$\frac{\text{bool_value} \equiv \text{mem_int}_1 \text{ binop}_{\text{rel}} \text{mem_int}_2}{\langle h; \text{mem_int}_1 \text{ binop}_{\text{rel}} \text{mem_int}_2 \rangle \longrightarrow \langle h; \text{done bool_value} \rangle} \quad \text{OP_MEMOP_TVAL_REL_BINOP}$$

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

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

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

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

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

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

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

$$\frac{}{\langle h + \{mem_ptr \mapsto_{\tau} pval\}; \mathbf{load}(\tau, mem_ptr, -, mem_ptr \mapsto_{\tau} pval) \rangle \longrightarrow \langle h + \{mem_ptr \mapsto_{\tau} pval\}; \mathbf{done} \, pval, mem_ptr \mapsto_{\tau} pval \rangle} \quad \text{OP_ACTION_TVAL_LOAD}$$

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

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

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

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

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

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

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

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

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

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

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

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

Definition rules: 213 good 0 bad
Definition rule clauses: 476 good 0 bad