

<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> $\text{array}(\overline{\text{loaded_value}_i}^i)$ $(\text{struct } \textit{ident})\{\overline{\text{member}_i:\tau_i = \text{mem_val}_i}^i\}$ $(\text{union } \textit{ident})\{\text{.member} = \text{mem_val}\}$	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
		$\text{error}(string, pval)$	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(⟦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>	$::=$ <i>ident</i> <i>pattern</i>	binders = <i>ident</i> binders = binders(<i>pattern</i>)
<i>bool_op</i>	$::=$ $\neg term$ $term_1 = term_2$ $term_1 \rightarrow term_2$ $\bigwedge(\overline{term_i}^i)$ $\bigvee(\overline{term_i}^i)$ $term_1 \text{ binop}_{bool} term_2$ if $term_1$ then $term_2$ else $term_3$	 M
<i>arith_op</i>	$::=$ $term_1 + term_2$ $term_1 - term_2$ $term_1 \times term_2$ $term_1 / term_2$ $term_1 \text{ rem_t } term_2$ $term_1 \text{ rem_f } term_2$ $term_1 \wedge term_2$ $term_1 \text{ binop}_{arith} term_2$	 M
<i>cmp_op</i>	$::=$ $term_1 < term_2$ $term_1 \leq term_2$ $term_1 \text{ binop}_{rel} term_2$	 less than less than or equal M
<i>list_op</i>	$::=$ nil	

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

$term, _$	$::=$		
		lit	
		$arith_op$	
		$bool_op$	
		cmp_op	
		$tuple_op$	
		$struct_op$	
		$pointer_op$	
		$list_op$	
		$array_op$	
		ct_pred	
		$param_op$	
		$(term)$	S parentheses
		$\sigma(term)$	M simul-sub σ in $term$
		$pval$	M
$pexpr$	$::=$		pure expressions
		$pval$	pure values
		$ctor_expr(\overline{pval}_i^i)$	data constructor application
		$array_shift(pval_1, \tau, pval_2)$	pointer array shift
		$member_shift(pval, ident, member)$	pointer struct/union member shift
		$not(pval)$	boolean not
		$pval_1 \ binop \ pval_2$	binary operations
		$memberof(ident, member, pval)$	C struct/union member access
		$name(\overline{pval}_i^i)$	pure function call
		$assert_undef(pval, UB_name)$	
		$bool_to_integer(pval)$	
		$conv_int(\tau, pval)$	
		$wrapI(\tau, pval)$	

$tpexpr$	$::=$ $ \quad tpval$ $ \quad \text{case } pval \text{ of } \overline{tpexpr_case_branch_i}^i \text{ end}$ $ \quad \text{let } ident_or_pattern = pexpr \text{ in } tpexpr$ $ \quad \text{let } ident_or_pattern:(y_1:\beta_1. term_1) = tpexpr_1 \text{ in } tpexpr_2$ $ \quad \text{if } pval \text{ then } tpexpr_1 \text{ else } tpexpr_2$ $ \quad \sigma(tpexpr)$	 $\text{bind binders}(ident_or_pattern) \text{ in } tpexpr$ $\text{bind binders}(ident_or_pattern) \text{ in } tpexpr_2$ $\text{bind } y_1 \text{ in } term_1$ M	top-level pure expressions top-level pure values pattern matching pure let annotated pure let pure if simul-sub σ in $tpexpr$
$tpexpr_case_branch$	$::=$ $ \quad pattern \Rightarrow tpexpr$	$\text{bind binders}(pattern) \text{ in } tpexpr$	pure top-level case expression top-level case expression br
m_kill_kind	$::=$ $ \quad \text{dynamic}$ $ \quad \text{static } \tau$		
$bool, _$	$::=$ $ \quad \text{true}$ $ \quad \text{false}$		OCaml booleans
$int, _$	$::=$ $ \quad i$		OCaml fixed-width integer literal integer
res_term	$::=$ $ \quad \text{emp}$ $ \quad points_to$ $ \quad ident$ $ \quad \langle res_term_1, res_term_2 \rangle$ $ \quad \text{pack}(pval, res_term)$ $ \quad \text{fold}(res_term)$		resource terms empty heap single-cell heap variable seperating-conjunction pair packing for existentials fold into recursive res. pred

	$\sigma(res_term)$	M	substitution for resource terms
<i>mem_action</i>	$::=$ create (<i>pval</i> , τ) create_readonly (<i>pval</i> ₁ , τ , <i>pval</i> ₂) alloc (<i>pval</i> ₁ , <i>pval</i> ₂) kill (<i>m_kill_kind</i> , <i>pval</i> , <i>pt</i>) store (<i>bool</i> , τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>mem_order</i> , <i>pt</i>) load (τ , <i>pval</i> , <i>mem_order</i> , <i>pt</i>) rmw (τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>pval</i> ₃ , <i>mem_order</i> ₁ , <i>mem_order</i> ₂) fence (<i>mem_order</i>) cmp_exch_strong (τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>pval</i> ₃ , <i>mem_order</i> ₁ , <i>mem_order</i> ₂) cmp_exch_weak (τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>pval</i> ₃ , <i>mem_order</i> ₁ , <i>mem_order</i> ₂) linux_fence (<i>linux_mem_order</i>) linux_load (τ , <i>pval</i> , <i>linux_mem_order</i>) linux_store (τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>linux_mem_order</i>) linux_rmw (τ , <i>pval</i> ₁ , <i>pval</i> ₂ , <i>linux_mem_order</i>)	memory actions	true means store is locking
<i>polarity</i>	$::=$ neg	polarities for memory actions	(pos) sequenced by let weak and let strong only sequenced by let strong
<i>pol_mem_action</i>	$::=$ <i>polarity mem_action</i>	memory actions with polarity	
<i>mem_op</i>	$::=$ <i>pval</i> ₁ <i>binop</i> _{rel} <i>pval</i> ₂ <i>pval</i> ₁ $-_{\tau}$ <i>pval</i> ₂ intFromPtr (τ_1 , τ_2 , <i>pval</i>) ptrFromInt (τ_1 , τ_2 , <i>pval</i>)	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

		<code>ptrValidForDeref</code> ($\tau, pval, pt$)		dereferencing validity predicate
		<code>ptrWellAligned</code> ($\tau, pval$)		
		<code>ptrArrayShift</code> ($pval_1, \tau, pval_2$)		
		<code>memcpy</code> ($pval_1, pval_2, pval_3$)		
		<code>memcmp</code> ($pval_1, pval_2, pval_3$)		
		<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> ($pval$)		
<i>spine_elem</i>	::=			spine element
		<i>pval</i>		pure or logical value
		<i>res_term</i>		resource value
		$\sigma(\textit{spine_elem})$	M	substitution for spine elements / return values
<i>spine</i>	::=			spine
		$\overline{\textit{spine_elem}_i}^i$		
<i>tval</i>	::=			(effectful) top-level values
		done <i>spine</i>		end of top-level expression
		undef <i>UB_name</i>		undefined behaviour
<i>res_pattern</i>	::=			resource terms
		emp	binders = {}	empty heap
		<i>ident</i>	binders = <i>ident</i>	variable
		fold (<i>res_pattern</i>)	binders = {}	unfold (recursive) predicate
		$\langle \textit{res_pattern}_1, \textit{res_pattern}_2 \rangle$	binders = binders(<i>res_pattern</i> ₁) \cup binders(<i>res_pattern</i> ₂)	seperating-conjunction pair
		pack (<i>ident, res_pattern</i>)	binders = <i>ident</i> \cup binders(<i>res_pattern</i>)	packing for existentials

$ret_pattern$	$::=$ $ \quad \mathbf{comp} \, ident_or_pattern$ $ \quad \mathbf{log} \, ident$ $ \quad \mathbf{res} \, res_pattern$	$binders = binders(ident_or_pattern)$ $binders = ident$ $binders = binders(res_pattern)$	return pattern computational variable logical variable resource variable
$init,$	$::=$ $ \quad \checkmark$ $ \quad \times$		initialisation status initialised uninitialised
$points_to, \, pt$	$::=$ $ \quad term_1 \xrightarrow{init}_\tau term_2$		points-to separation logic predicate
res	$::=$ $ \quad \mathbf{emp}$ $ \quad points_to$ $ \quad res_1 * res_2$ $ \quad \exists ident:\beta. res$ $ \quad term \wedge res$ $ \quad \mathbf{if} \, term \, \mathbf{then} \, res_1 \, \mathbf{else} \, res_2$ $ \quad \alpha(\overline{pval_i}^i)$ $ \quad \sigma(res)$	 M	resources empty heap points-top heap pred. seperating conjunction existential logical conjunction ordered disjunction predicate simul-sub σ in res
$ret, \, _$	$::=$ $ \quad \Sigma ident:\beta. ret$ $ \quad \exists ident:\beta. ret$ $ \quad res \otimes ret$ $ \quad term \wedge ret$ $ \quad \mathbf{I}$ $ \quad \sigma(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

<i>seq_expr</i>	$::=$ ccall ($\tau, ident, spine$) pcall ($name, spine$)		sequential (effectful) expressions C function call procedure call
<i>seq_texpr</i>	$::=$ <i>tval</i> run $ident \overline{pval}_i^i$ let $ident_or_pattern = pexpr$ in <i>texpr</i> let $ident_or_pattern:(y_1:\beta_1. term_1) = tpexpr$ in <i>texpr</i> let $\overline{ret_pattern}_i^i = seq_expr$ in <i>texpr</i> let $\overline{ret_pattern}_i^i:ret = texpr_1$ in <i>texpr</i> ₂ case <i>pval</i> of $\overline{texpr_case_branch}_i^i$ end if <i>pval</i> then <i>texpr</i> ₁ else <i>texpr</i> ₂ bound [<i>int</i>] (<i>is_texpr</i>)	bind binders(<i>ident_or_pattern</i>) in <i>texpr</i> bind binders(<i>ident_or_pattern</i>) in <i>texpr</i> bind y_1 in $term_1$ bind binders($\overline{ret_pattern}_i^i$) in <i>texpr</i> bind binders($\overline{ret_pattern}_i^i$) in <i>texpr</i> 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 limit scope of indet seq behaviour
<i>texpr_case_branch</i>	$::=$ $pattern \Rightarrow texpr$	bind binders(<i>pattern</i>) in <i>texpr</i>	top-level case expression branch top-level case expression branch
<i>is_expr</i>	$::=$ <i>tval</i> memop (<i>mem_op</i>) <i>pol_mem_action</i>		indet seq (effectful) expressions (effectful) top-level values pointer op involving memory memory action
<i>is_texpr</i>	$::=$ let weak $\overline{ret_pattern}_i^i = is_expr$ in <i>texpr</i> let strong $\overline{ret_pattern}_i^i = is_expr$ in <i>texpr</i>	bind binders($\overline{ret_pattern}_i^i$) in <i>texpr</i> bind binders($\overline{ret_pattern}_i^i$) in <i>texpr</i>	indet seq top-level (effectful) expressions weak sequencing strong sequencing
<i>texpr</i>	$::=$ <i>seq_texpr</i>		top-level (effectful) expressions sequential (effectful) expressions

		is_expr		indet seq (effectful) expressions
		$\sigma(expr)$	M	simul-sub σ in $expr$
arg	::=			argument/function types
		$\Pi ident:\beta. arg$		
		$\forall ident:\beta. arg$		
		$res \multimap arg$		
		$term \supset arg$		
		ret		
		$\sigma(arg)$	M	simul-sub σ in arg
$pure_arg$::=			pure argument/function types
		$\Pi ident:\beta. pure_arg$		
		$term \supset pure_arg$		
		$pure_ret$		
$pure_ret$::=			pure return types
		$\Sigma ident:\beta. pure_ret$		
		$term \wedge pure_ret$		
		\mathbf{I}		
\mathcal{C}	::=			computational var env
		\cdot		
		$\mathcal{C}, ident:\beta$		
		$\overline{\mathcal{C}}_i^i$		
\mathcal{L}	::=			logical var env
		\cdot		
		$\overline{\mathcal{L}}_i^i$		
		$\mathcal{L}, ident:\beta$		

Φ	$::=$ \mid $\mid \quad \cdot$ $\mid \quad \Phi, term$ $\mid \quad \overline{\Phi_i}^i$	constraints env
\mathcal{R}	$::=$ \mid $\mid \quad \cdot$ $\mid \quad \mathcal{R}, ident:res$ $\mid \quad \overline{\mathcal{R}_i}^i$	resources env
σ, ψ	$::=$ \mid $\mid \quad \cdot$ $\mid \quad spine_elem/ident, \sigma$ $\mid \quad term/ident, \sigma$ $\mid \quad res/ident, \sigma$ $\mid \quad \overline{\sigma_i}^i$ $\mid \quad \sigma(\psi)$	substitutions M apply σ to all elements in ψ
$typing$	$::=$ \mid $\mid \quad \mathbf{smt}(\Phi \Rightarrow term)$ $\mid \quad ident:\beta \in \mathcal{C}$ $\mid \quad ident:\beta \in \mathcal{L}$ $\mid \quad \mathbf{struct} tag \ \& \ \overline{member_i:\tau_i}^i \in \mathbf{Globals}$ $\mid \quad \alpha \equiv \overline{x_i:\beta_i}^i \mapsto res \in \mathbf{Globals}$ $\mid \quad \overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash mem_val_i \Rightarrow \mathbf{mem} \beta_i}^i$	 recursive resource predicate dependent on memory object model
$opsem$	$::=$ \mid $\mid \quad \forall i < j. \mathbf{not} (pattern_i = pval \rightsquigarrow \sigma_i)$ $\mid \quad \mathbf{fresh}(mem_ptr)$ $\mid \quad term$	

		$pval;\beta$	
$formula$	$::=$	$judgement$ $typing$ $opsem$ $term \equiv term'$ $name: pure_arg \equiv \overline{x_i}^i \mapsto tpepr \in \mathbf{Globals}$ $name: arg \equiv \overline{x_i}^i \mapsto texpr \in \mathbf{Globals}$	
$heap, h$	$::=$		heaps
		.	
		$h + \{points_to\}$	
$lemma_jtype$	$::=$	$\overline{x_i}^i :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma):(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$	
res_jtype	$::=$	$\Phi \vdash res \equiv res'$ $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow res$ $h:\mathcal{R}$	
$object_value_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi \vdash object_value \Rightarrow \mathbf{obj} \beta$	
$pval_jtype$	$::=$	$\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta$	

$spine_jtype$	$::=$ $ \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} \text{ with } term$ $ \quad ident_or_pattern:\beta \rightsquigarrow \mathcal{C} \text{ with } term$
$res_pattern_jtype$	$::=$ $ \quad \mathcal{L}; \Phi \vdash res' = strip_ifs(res)$ $ \quad \mathcal{L}; \Phi \vdash res \text{ as } res_pattern \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'$ $ \quad \mathcal{L}; \Phi \vdash res_pattern:res \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'$
$ret_pattern_jtype$	$::=$ $ \quad \mathcal{L}; \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 \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \sigma$ $ \quad \overline{x_i = spine_elem_i}^i :: arg \gg \sigma; ret$
$pure_opsem_jtype$	$::=$	$ \quad \langle pexpr \rangle \longrightarrow \langle pexpr' \rangle$ $ \quad \langle pexpr \rangle \longrightarrow \langle texpr:(y:\beta. term) \rangle$ $ \quad \langle texpr \rangle \longrightarrow \langle texpr' \rangle$
$opsem_jtype$	$::=$	$ \quad \langle h; seq_expr \rangle \longrightarrow \langle h'; texpr:ret \rangle$ $ \quad \langle h; seq_texpr \rangle \longrightarrow \langle h'; texpr \rangle$ $ \quad \langle h; mem_op \rangle \longrightarrow \langle h'; tval \rangle$

$$\begin{array}{l}
| \langle h; mem_action \rangle \longrightarrow \langle h'; tval \rangle \\
| \langle h; is_expr \rangle \longrightarrow \langle h'; is_expr' \rangle \\
| \langle h; is_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
| \langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle
\end{array}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\Phi \vdash res_1 \equiv res'_1 \quad \Phi \vdash res_2 \equiv res'_2}{\Phi \vdash res_1 * res_2 \equiv res'_1 * res'_2} \quad \text{TY_RES_EQ_SEP_CONJ}$$

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

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

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

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

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

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

$$\frac{\begin{array}{l} \Phi \vdash points_to \equiv points_to' \\ \Phi \vdash points_to' \equiv points_to'' \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, \cdot, points_to \vdash points_to' \Leftarrow points_to''} \text{TY_RES_POINTSTO}$$

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

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

$$\frac{\begin{array}{l} \mathbf{smt}(\Phi \Rightarrow term) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow res \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res_term \Leftarrow term \wedge res} \text{TY_RES_CONJ}$$

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

$$\begin{array}{c}
\alpha \equiv \overline{x_i:\beta_i}^i \mapsto res \in \mathbf{Globals} \\
\overline{\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow \beta_i}^i \\
\mathcal{L};\Phi \vdash res' = \mathbf{strip_ifs}(\overline{pval_i/x_i, \cdot}^i(res)) \\
\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash res_term \Leftarrow res' \\
\hline
\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathbf{fold}(res_term) \Leftarrow \alpha(\overline{pval_i}^i) \quad \text{TY_RES_FOLD}
\end{array}$$

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

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

$$\begin{array}{c}
h:\mathcal{R} \\
\vdots; \vdots; \mathcal{R}' \vdash pt \Leftarrow pt \\
\hline
h + \{pt\}:\mathcal{R}, \mathcal{R}' \quad \text{TY_HEAP_POINTSTO}
\end{array}$$

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

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

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

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

$$\begin{array}{c}
\mathbf{struct} \mathbf{tag} \ \& \ \overline{member_i:\tau_i}^i \in \mathbf{Globals} \\
\overline{\mathcal{C};\mathcal{L};\Phi \vdash mem_val_i \Rightarrow \mathbf{mem} \beta_{\tau_i}}^i \\
\hline
\mathcal{C};\mathcal{L};\Phi \vdash (\mathbf{struct} \mathbf{tag})\{\overline{member_i:\tau_i}^i = mem_val_i^i\} \Rightarrow \mathbf{obj} \mathbf{struct} \mathbf{tag} \quad \text{TY_PVAL_OBJ_STRUCT}
\end{array}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\text{pattern}_1:\beta \rightsquigarrow \mathcal{C}_1 \text{ with } term_1 \quad \text{pattern}_2:\text{list } \beta \rightsquigarrow \mathcal{C}_2 \text{ with } term_2}{\text{Cons}(\text{pattern}_1, \text{pattern}_2):\text{list } \beta \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2 \text{ with } term_1 :: term_2} \quad \text{TY_PAT_COMP_CONS}$$

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

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

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

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

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

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

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

$$\frac{}{\mathcal{L}; \Phi \vdash \text{emp} = \text{strip_ifs}(\text{emp})} \text{TY_PAT_RES_STRIP_IFS_EMPTY}$$

$$\frac{}{\mathcal{L}; \Phi \vdash pt = \text{strip_ifs}(pt)} \text{TY_PAT_RES_STRIP_IFS_POINTSTO}$$

$$\frac{}{\mathcal{L}; \Phi \vdash res_1 * res_2 = \text{strip_ifs}(res_1 * res_2)} \text{TY_PAT_RES_STRIP_IFS_SEP_CONJ}$$

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

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

$$\frac{\begin{array}{c} \text{smt}(\Phi \Rightarrow term) \\ \mathcal{L}; \Phi \vdash res'_1 = \text{strip_ifs}(res'_1) \end{array}}{\mathcal{L}; \Phi \vdash res'_1 = \text{strip_ifs}(\text{if } term \text{ then } res_1 \text{ else } res_2)} \text{TY_PAT_RES_STRIP_IFS_TRUE}$$

$$\frac{\begin{array}{c} \text{smt}(\Phi \Rightarrow \neg term) \\ \mathcal{L}; \Phi \vdash res'_2 = \text{strip_ifs}(res_2) \end{array}}{\mathcal{L}; \Phi \vdash res'_2 = \text{strip_ifs}(\text{if } term \text{ then } res_1 \text{ else } res_2)} \text{TY_PAT_RES_STRIP_IFS_FALSE}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool} \\ \mathcal{C}; \mathcal{L}; \Phi, pval = \text{true} \vdash tpexpr_1 \Leftarrow y:\beta. term \\ \mathcal{C}; \mathcal{L}; \Phi, pval = \text{false} \vdash tpexpr_2 \Leftarrow y:\beta. term \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \textbf{if } pval \textbf{ then } tpexpr_1 \textbf{ else } tpexpr_2 \Leftarrow y:\beta. term} \quad \text{TY_TPE_IF}$$

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

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash tpepr_1 \Leftarrow y_1:\beta_1. term_1 \\
ident_or_pattern:\beta_1 \rightsquigarrow \mathcal{C}_1 \text{ with } term \\
\mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term/y_1, \cdot(term_1) \vdash tpepr \Leftarrow y_2:\beta_2. term_2 \\
\hline
\mathcal{C}; \mathcal{L}; \Phi \vdash \text{let } ident_or_pattern:(y_1:\beta_1. term_1) = tpepr_1 \text{ in } tpepr_2 \Leftarrow y_2:\beta_2. term_2
\end{array}
\quad \text{TY_TPE_LETT}$$

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

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

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{integer} \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{create}(pval, \tau) \Rightarrow \Sigma y_p:\text{loc. representable}(\tau*, y_p) \wedge \text{alignedI}(pval, y_p) \wedge \exists y:\beta_\tau. y_p \overset{\times}{\mapsto}_\tau y \otimes \text{I}
\end{array}
\quad \text{TY_ACTION_CREATE}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{array}{c}
\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \text{bool} \\
\mathcal{C}; \mathcal{L}; \Phi, pval = \text{true}; \mathcal{R} \vdash texpr_1 \Leftarrow ret \\
\mathcal{C}; \mathcal{L}; \Phi, pval = \text{false}; \mathcal{R} \vdash texpr_2 \Leftarrow ret \\
\hline
\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{if } pval \text{ then } texpr_1 \text{ else } texpr_2 \Leftarrow ret
\end{array}
\quad \text{TY_SEQ_TE_IF}$$

$$\frac{\text{ident:arg} \equiv \overline{x_i}^i \mapsto \text{texpr} \in \text{Globals} \quad \mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \overline{x_i = \text{pval}_i^i}^i :: \text{arg} \gg \sigma; \text{false} \wedge \text{I}}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \text{run ident } \overline{\text{pval}_i^i}^i \Leftarrow \text{false} \wedge \text{I}} \quad \text{TY_SEQ_TE_RUN}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_texpr} \Leftarrow \text{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{bound}[\text{int}](\text{is_texpr}) \Leftarrow \text{ret}} \quad \text{TY_SEQ_TE_BOUND}$$

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

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

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_texpr} \Leftarrow \text{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{is_texpr} \Leftarrow \text{ret}} \quad \text{TY_TE_IS}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{seq_texpr} \Leftarrow \text{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \text{seq_texpr} \Leftarrow \text{ret}} \quad \text{TY_TE_SEQ}$$

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

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

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

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

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

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

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

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

$$\frac{}{x = pval \rightsquigarrow pval/x, \cdot} \quad \text{SUBS_DECONS_VALUE_SYM}$$

$$\frac{pattern = pval \rightsquigarrow \sigma}{pattern = pval \rightsquigarrow \sigma} \quad \text{SUBS_DECONS_VALUE_PATTERN}$$

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

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

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

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

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

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

$$\boxed{\overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \sigma}$$

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

$$\frac{\begin{array}{l} ident_or_pattern = pval \rightsquigarrow \sigma \\ \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \psi \end{array}}{\mathbf{comp} ident_or_pattern = pval, \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \sigma, \psi} \quad \text{SUBS_DECONS_RET_COMP}$$

$$\frac{\overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \psi}{\mathbf{log} ident = pval, \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow pval/ident, \psi} \quad \text{SUBS_DECONS_RET_LOG}$$

$$\frac{\begin{array}{l} res_pattern = res_term \rightsquigarrow \sigma \\ \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \psi \end{array}}{\mathbf{res} res_pattern = res_term, \overline{ret_pattern_i = spine_elem_i}^i \rightsquigarrow \sigma, \psi} \quad \text{SUBS_DECONS_RET_RES}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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