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

Sctypes_t, τ	::=			C type
	|	τ*		pointer to type τ
tag	::=			OCaml type for struct/union tag
	|	*ident*		
β, _	::=			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

	`>`	greater than
	`<`	less than
	`>=`	greater than or equal to
	`<=`	less than or equal to
	`/\`	conjucntion
	`\/`	disjunction

$binop_{arith}$	::=		arithmentic binary operators
	|	`+`	
	|	`-`	
	|	`*`	
	|	`/`	
	|	`rem_t`	
	|	`rem_f`	
	|	`^`	

$binop_{rel}$	::=		relational binary operators
	|	`=`	
	|	`>`	
	|	`<`	
	|	`>=`	
	|	`<=`	

$binop_{bool}$	::=		boolean binary operators
	|	`/\`	
	|	`\/`	

mem_int	::=			memory integer value
	|	1	M	
	|	0	M	

object_value	::=		C object values (inhabitants of object types), which can be read/stored
	|	*mem_int*	integer value
	|	*mem_ptr*	pointer value
	|	`array` $(\overline{loaded\_value_i}^{\,i})$	C array value
	|	$(\texttt{struct}\, ident)\{\overline{.member_i{:}\tau_i = mem\_val_i}^{\,i}\}$	C struct value
	|	$(\texttt{union}\, ident)\{.member = mem\_val\}$	C union value
loaded_value	::=		potentially unspecified C object values
	|	`specified` *object_value*	specified loaded value
value	::=		Core values
	|	*object_value*	C object value
	|	*loaded_value*	loaded C object value
	|	`Unit`	unit
	|	`True`	boolean true
	|	`False`	boolean false
	|	$\beta[\overline{value_i}^{\,i}]$	list
	|	$(\overline{value_i}^{\,i})$	tuple
bool_value	::=		Core booleans
	|	`True`	boolean true
	|	`False`	boolean false
ctor_val	::=		data constructors
	|	`Nil` β	empty list
	|	`Cons`	list cons
	|	`Tuple`	tuple
	|	`Array`	C array
	|	`Specified`	non-unspecified loaded value

ctor_expr	::=		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
name	::=		
	|	*ident*	Core identifier
	|	*impl_const*	implementation-defined constant
pval	::=		pure values
	|	*ident*	Core identifier
	|	*impl_const*	implementation-defined constant
	|	*value*	Core values
	|	`constrained` $(\overline{mem\_iv\_c_i, pval_i}^{\,i})$	constrained value
	|	`error` $(string, pval)$	impl-defined static error
	|	$ctor\_val(\overline{pval_i}^{\,i})$	data constructor application
	|	$(\texttt{struct}\, ident)\{\,\overline{.member_i = pval_i}^{\,i}\,\}$	C struct expression
	|	$(\texttt{union}\, ident)\{\,.member = pval\}$	C union expression
tpval	::=		top-level pure values
	|	`undef` *UB_name*	undefined behaviour
	|	`done` *pval*	pure done

ident_opt_β	::=			type annotated optional identifier
	|	$\_{:}\beta$	binders = {}	
	|	*ident*:β	binders = *ident*	
pattern	::=			
	|	*ident_opt_β*	binders = binders(*ident_opt_β*)	
	|	*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(τ)	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$_\tau$	M	maximum value of int of type τ
	|	min_int$_\tau$	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 ::=

|	$\neg\, term$	
|	$term_1 = term_2$	
|	$\bigwedge(\overline{term_i}^{\,i})$	
|	$\bigvee(\overline{term_i}^{\,i})$	
|	$term_1\ binop_{bool}\ term_2$	M
|	$\texttt{if}\ term_1\ \texttt{then}\ term_2\ \texttt{else}\ term_3$	

arith_op ::=

|	$term_1 + term_2$	
|	$term_1 - term_2$	
|	$term_1 \times term_2$	
|	$term_1 / term_2$	
|	$term_1\ \texttt{rem\_t}\ term_2$	
|	$term_1\ \texttt{rem\_f}\ term_2$	
|	$term_1\ \texttt{\^{}}\ term_2$	
|	$term_1\ binop_{arith}\ term_2$	M

cmp_op ::=

|	$term_1 < term_2$		less than
|	$term_1 \leq term_2$		less than or equal
|	$term_1\ binop_{rel}\ term_2$	M	

list_op ::=

|	$\texttt{nil}$	
|	$\texttt{tl}\ term$	
|	$term^{(int)}$	

tuple_op ::=

|	$(\overline{term_i}^{\,i})$	

$$
\begin{array}{lll}
 & | & term^{(int)} \\
 & & \\
pointer\_op & ::= & \\
 & | & mem\_ptr \\
 & | & term_1 +_{\mathrm{ptr}} term_2 \\
 & | & \texttt{cast\_int\_to\_ptr}\, term \\
 & | & \texttt{cast\_ptr\_to\_int}\, term \\
 & & \\
array\_op & ::= & \\
 & | & term_1[term_2] \\
 & & \\
param\_op & ::= & \\
 & | & ident{:}\beta.\, term \\
 & | & term(term_1, \ldots, term_n) \\
 & & \\
struct\_op & ::= & \\
 & | & term.member \\
 & & \\
ct\_pred & ::= & \\
 & | & \texttt{representable}\,(\tau, term) \\
 & | & \texttt{aligned}\,(\tau, term) \\
 & | & \texttt{alignedI}\,(term_1, term_2) \\
 & & \\
term,\ \_ & ::= & \\
 & | & lit \\
 & | & arith\_op \\
 & | & bool\_op \\
 & | & cmp\_op \\
 & | & tuple\_op \\
 & | & struct\_op
\end{array}
$$

	$pointer\_op$		
	$list\_op$		
	$array\_op$		
	ct_pred		
	`option_op`		
	$param\_op$		
	$(term)$	S	parentheses
	$\sigma(term)$	M	
	$[term_1/ident]term_2$	M	substitute $term_1$ for $ident$ in $term_2$
	$pval$	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` $(\tau, pval)$			

$tpexpr$	::=			top-level pure expressions
	$tpval$			top-level pure values
	`case` $pval$ `of` $\overline{\mid\ tpexpr\_case\_branch_i}^{\,i}$ `end`			pattern matching
	`let` $ident\_or\_pattern = pexpr$ `in` $tpexpr$	bind binders($ident\_or\_pattern$) in $tpexpr$		pure let
	`let` $ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr_1$ `in` $tpexpr_2$	bind binders($ident\_or\_pattern$) in $tpexpr_2$		pure let

			bind y_1 in $term_1$	
	|	`if` $pval$ `then` $tpexpr_1$ `else` $tpexpr_2$		pure if
	|	$\sigma(tpexpr)$	M	simul-sub environment σ in
$tpexpr\_case\_branch$	::=			pure top-level case expression
	|	$pattern \Rightarrow tpexpr$	bind binders($pattern$) in $tpexpr$	top-level case expression br
m_kill_kind	::=			
	|	`dynamic`		
	|	`static` τ		
$bool$, _	::=			OCaml booleans
	|	`true`		
	|	`false`		
int, _	::=			OCaml fixed-width integer
	|	i		literal integer
mem_action	::=			memory actions
	|	`create` $(pval, \tau)$		
	|	`create_readonly` $(pval_1, \tau, pval_2)$		
	|	`alloc` $(pval_1, pval_2)$		
	|	`kill` $(m\_kill\_kind, pval)$		
	|	`store` $(bool, \tau, pval_1, pval_2, mem\_order)$		true means store is locking
	|	`load` $(\tau, pval, mem\_order)$		
	|	`rmw` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`fence` (mem_order)		
	|	`cmp_exch_strong` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`cmp_exch_weak` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`linux_fence` $(linux\_mem\_order)$		

	|	$\texttt{linux\_load}\,(\tau, pval, linux\_mem\_order)$	
	|	$\texttt{linux\_store}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	
	|	$\texttt{linux\_rmw}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	
polarity	::=		polarities for memory actions
	|		(pos) sequenced by `let weak` and `let strong`
	|	`neg`	only sequenced by `let strong`
pol_mem_action	::=		memory actions with polarity
	|	*polarity mem_action*	
mem_op	::=		operations involving the memory state
	|	$pval_1 == pval_2$	pointer equality comparison
	|	$pval_1 \neq pval_2$	pointer inequality comparison
	|	$pval_1 < pval_2$	pointer less-than comparison
	|	$pval_1 > pval_2$	pointer greater-than comparison
	|	$pval_1 \leq pval_2$	pointer less-than comparison
	|	$pval_1 \geq pval_2$	pointer greater-than comparison
	|	$pval_1 -_\tau pval_2$	pointer subtraction
	|	$\texttt{intFromPtr}\,(\tau_1, \tau_2, pval)$	cast of pointer value to integer value
	|	$\texttt{ptrFromInt}\,(\tau_1, \tau_2, pval)$	cast of integer value to pointer value
	|	$\texttt{ptrValidForDeref}\,(\tau, pval)$	dereferencing validity predicate
	|	$\texttt{ptrWellAligned}\,(\tau, pval)$	
	|	$\texttt{ptrArrayShift}\,(pval_1, \tau, pval_2)$	
	|	$\texttt{memcpy}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{memcmp}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{realloc}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{va\_start}\,(pval_1, pval_2)$	
	|	$\texttt{va\_copy}\,(pval)$	
	|	$\texttt{va\_arg}\,(pval, \tau)$	

	$\mid$	$\texttt{va\_end}\,(pval)$	
res_term	::=		resource terms
	$\mid$	$\texttt{emp}$	empty heap
	$\mid$	$\texttt{pt}$	single-cell heap
	$\mid$	*ident*	variable
	$\mid$	$\langle res\_term_1, res\_term_2 \rangle$	seperating-conjunction pair
	$\mid$	$\texttt{pack}\,(pval, res\_term)$	packing for existentials
spine_elem	::=		spine element
	$\mid$	*pval*	pure value
	$\mid$	*term*	logical value
	$\mid$	*res_term*	resource value
spine	::=		spine
	$\mid$	$\overline{spine\_elem_i}^{\,i}$	
tval	::=		(effectful) top-level values
	$\mid$	$\texttt{done}\ spine$	end of top-level expression
	$\mid$	$\texttt{undef}\ UB\_name$	undefined behaviour
res_pattern	::=		resource terms
	$\mid$	$\texttt{emp}$ binders $= \{\}$	empty heap
	$\mid$	$\texttt{pt}$ binders $= \{\}$	single-cell heap
	$\mid$	*ident* binders $= ident$	variable
	$\mid$	$\langle res\_pattern_1, res\_pattern_2 \rangle$ binders $=$ binders$(res_pattern_1) \cup$ binders$(res_pattern_2)$	seperating-conjunction pair
	$\mid$	$\texttt{pack}\,(ident, res\_pattern)$ binders $= ident \cup$ binders$(res_pattern)$	packing for existentials
ret_pattern	::=		return pattern
	$\mid$	$\texttt{comp}\ ident\_or\_pattern$ binders $=$ binders$(ident_or_pattern)$	computational variable

	$\texttt{log}\,ident$	binders = $ident$	logical variable
	$res\ res\_pattern$	binders = binders($res\_pattern$)	resource variable
$init,$	::=		initialisation status
	✓		initialised
	×		uninitalised
$points\_to$	::=		points-to separation logic predicate
	$term_1\ \mathbb{Q}^{init}_{\mapsto\tau}\ term_2$		
res	::=		resources
	$\texttt{emp}$		empty heap
	$points\_to$		points-top heap pred.
	$res_1 * res_2$		seperating conjunction
	$\exists\, ident{:}\beta.\ res$		existential
	$term \wedge res$		logical conjuction
	$\langle res \rangle$	S	parentheses
	$[pval/ident]res$	M	substitute $pval$ for $ident$ in res
$ret,\ \_$	::=		return types
	$\Sigma\, ident{:}\beta.\ ret$		
	$\exists\, ident{:}\beta.\ ret$		
	$res \otimes ret$		
	$term \wedge ret$		
	$\texttt{I}$		
	$[spine\_elem/ident]ret$	M	
	$\sigma(ret)$	M	
seq_expr	::=		sequential (effectful) expressions
	$\texttt{ccall}\,(\tau, pval, spine)$		C function call

	$\mid$ `pcall` $(name, spine)$		procedure call
seq_texpr	::=		sequential top-level (effectful) expres
	$\mid$ $tval$		(effectful) top-level values
	$\mid$ `run` $ident\ \overline{pval_i}^{\,i}$		run from label
	$\mid$ `let` $ident\_or\_pattern = pexpr$ `in` $texpr$	bind binders($ident\_or\_pattern$) in $texpr$	pure let
	$\mid$ `let` $ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr$ `in` $texpr$	bind binders($ident\_or\_pattern$) in $texpr$ bind y_1 in $term_1$	pure let
	$\mid$ `let` $\overline{ret\_pattern_i}^{\,i} = seq\_expr$ `in` $texpr$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr$	bind return patterns
	$\mid$ `let` $\overline{ret\_pattern_i}^{\,i} {:} ret = texpr_1$ `in` $texpr_2$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr_2$	annotated bind return patterns
	$\mid$ `case` $pval$ `of` $\overline{\mid\ texpr\_case\_branch_i}^{\,i}$ `end`		pattern matching
	$\mid$ `if` $pval$ `then` $texpr_1$ `else` $texpr_2$		conditional
	$\mid$ `bound` $[int](is\_texpr)$		limit scope of indet seq behaviour
$texpr\_case\_branch$	::=		top-level case expression branch
	$\mid$ $pattern \Rightarrow texpr$	bind binders($pattern$) in $texpr$	top-level case expression branch
is_expr	::=		indet seq (effectful) expressions
	$\mid$ $tval$		(effectful) top-level values
	$\mid$ `memop` (mem_op)		pointer op involving memory
	$\mid$ pol_mem_action		memory action
is_texpr	::=		indet seq top-level (effectful) expres
	$\mid$ `let weak` $\overline{ret\_pattern_i}^{\,i} = is\_expr$ `in` $texpr$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr$	weak sequencing
	$\mid$ `let strong` $\overline{ret\_pattern_i}^{\,i} = is\_expr$ `in` $texpr$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr$	strong sequencing
$texpr$	::=		top-level (effectful) expressions
	$\mid$ seq_texpr		sequential (effectful) expressions
	$\mid$ is_texpr		indet seq (effectful) expressions
	$\mid$ $\sigma(texpr)$	M	simul-sub environment σ in $texpr$

arg	::=		argument/function types
	|	$\Pi\, ident{:}\beta.\ arg$	
	|	$\forall\, ident{:}\beta.\ arg$	
	|	$res \multimap arg$	
	|	$term \supset arg$	
	|	ret	
	|	$[spine\_elem/ident]arg$	M
$pure\_arg$	::=		pure argument/function types
	|	$\Pi\, ident{:}\beta.\ pure\_arg$	
	|	$term \supset pure\_arg$	
	|	$pure\_ret$	
$pure\_ret$	::=		pure return types
	|	$\Sigma\, ident{:}\beta.\ pure\_ret$	
	|	$term \wedge pure\_ret$	
	|	$\mathtt{I}$	
$\mathcal{C}$	::=		computational var env
	|	$.$	
	|	$\mathcal{C}, ident{:}\beta$	
	|	$\overline{\mathcal{C}_i}^{\,i}$	
$\mathcal{L}$	::=		logical var env
	|	$.$	
	|	$\overline{\mathcal{L}_i}^{\,i}$	
	|	$\mathcal{L}, ident{:}\beta$	
Φ	::=		constraints env
	|	$.$	

	$\mid$	$\Phi, term$
	$\mid$	$\overline{\Phi_i}^{\,i}$
$\mathcal{R}$	::=	resources env
	$\mid$	$.$
	$\mid$	$\mathcal{R}, ident{:}res$
	$\mid$	$\overline{\mathcal{R}_i}^{\,i}$
σ	::=	values env
	$\mid$	$.$
	$\mid$	$spine\_elem/ident, \sigma$
	$\mid$	$\overline{\sigma_i}^{\,i}$
typing	::=	
	$\mid$	$\texttt{smt}\,(\Phi \Rightarrow term)$
	$\mid$	$ident{:}\beta \in \mathcal{C}$
	$\mid$	$ident{:}\,\texttt{struct}\,tag\ \&\,\overline{member_i{:}\tau_i}^{\,i} \in \texttt{Globals}$
	$\mid$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash mem\_val_i \Rightarrow \texttt{mem}\,\beta_i}^{\,i}$ dependent on memory object model
	$\mid$	$\mathcal{L} \vdash term{:}\beta$
	$\mid$	$pval{:}arg \in \texttt{Globals}$
opsem	::=	
	$\mid$	$pval \equiv \overline{x_i}^{\,i} \mapsto tpexpr{:}(y{:}\beta.\ term) \in \texttt{Globals}$
	$\mid$	$pval \equiv \overline{x_i}^{\,i} \mapsto texpr{:}ret \in \texttt{Globals}$
	$\mid$	$\forall\, i < j.\ \texttt{not}\,(pattern_i = pval \rightsquigarrow \sigma_i)$
	$\mid$	$\texttt{fresh}\,(mem\_ptr)$
formula	::=	
	$\mid$	$judgement$
	$\mid$	$term \equiv term'$

	|	$typing$
	|	$opsem$
$heap,\ h$	::=	heaps
	|	$h + \{mem\_ptr \mapsto \times\}$
	|	$h + \{mem\_ptr \mapsto pval\}$
$object\_value\_jtype$	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash object\_value \Rightarrow \texttt{obj}\ \beta$
$pval\_jtype$	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta$
res_jtype	::=	
	|	$\Phi \vdash res \equiv res'$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Leftarrow res$
$spine\_jtype$	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: arg \gg ret$
$pexpr\_jtype$	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow ident{:}\beta.\ term$
$comp\_pattern\_jtype$	::=	
	|	$term\ \texttt{as}\ pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi$
	|	$term\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi$
$res\_pattern\_jtype$	::=	
	|	$res\_pattern{:}res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}$

$$
\begin{array}{lll}
\mathit{ret\_pattern\_jtype} & ::= & \\
 & \mid & \overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \\
 & & \\
\mathit{tpval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{action\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{tval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{texpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret} \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{decons\_jtype} & ::= & \\
 & \mid & \mathit{pattern} = \mathit{pval} \rightsquigarrow \sigma \\
 & \mid & \mathit{ident\_or\_pattern} = \mathit{pval} \rightsquigarrow \sigma \\
 & \mid & \mathit{res\_pattern} = \mathit{res\_term} \rightsquigarrow \sigma \\
 & \mid & \mathit{ret\_pattern} = \mathit{spine\_elem} \rightsquigarrow \sigma \\
 & & \\
\mathit{pure\_opsem\_jtype} & ::= & \\
 & \mid & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle
\end{array}
$$

$$
\begin{array}{lll}
 & | & \langle pexpr \rangle \longrightarrow \langle tpexpr{:}(y{:}\beta.\ term) \rangle \\
 & | & \langle tpexpr \rangle \longrightarrow \langle tpexpr' \rangle
\end{array}
$$

$$
\begin{array}{lll}
opsem\_jtype & ::= & \\
 & | & \langle seq\_expr \rangle \longrightarrow \langle texpr{:}ret \rangle \\
 & | & \langle h; seq\_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
 & | & \langle mem\_op \rangle \longrightarrow \langle pval \rangle \\
 & | & \langle h; mem\_action \rangle \longrightarrow \langle h'; spine \rangle \\
 & | & \langle h; is\_expr \rangle \longrightarrow \langle h'; is\_expr' \rangle \\
 & | & \langle h; is\_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
 & | & \langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle
\end{array}
$$

$\boxed{\mathcal{C}; \mathcal{L}; \Phi \vdash object\_value \Rightarrow \texttt{obj}\ \beta}$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash mem\_int \Rightarrow \texttt{obj integer}} \quad \textsc{Ty\_Pval\_Obj\_Int}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash mem\_ptr \Rightarrow \texttt{obj loc}} \quad \textsc{Ty\_Pval\_Obj\_Ptr}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash loaded\_value_i \Rightarrow \beta}^{\,i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{array}\,(\,\overline{loaded\_value_i}^{\,i}\,) \Rightarrow \texttt{obj array}\,\beta} \quad \textsc{Ty\_Pval\_Obj\_Arr}$$

$$\frac{\begin{array}{l} ident{:}\,\texttt{struct}\,tag\ \&\ \overline{member_i{:}\tau_i}^{\,i} \in \texttt{Globals} \\ \overline{\mathcal{C}; \mathcal{L}; \Phi \vdash mem\_val_i \Rightarrow \texttt{mem}\,\beta_i}^{\,i} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash (\,\texttt{struct}\,tag)\{\,\overline{.member_i{:}\tau_i = mem\_val_i}^{\,i}\,\} \Rightarrow \texttt{obj struct}\,tag} \quad \textsc{Ty\_Pval\_Obj\_Struct}$$

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

$$\frac{x{:}\beta \in \mathcal{C}}{\mathcal{C}; \mathcal{L}; \Phi \vdash x \Rightarrow \beta} \quad \textsc{Ty\_Pval\_Var}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{object\_value} \Rightarrow \texttt{obj}\,\beta}{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{object\_value} \Rightarrow \beta}\ \textsc{Ty\_Pval\_Obj}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{object\_value} \Rightarrow \texttt{obj}\,\beta}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{specified}\,\mathit{object\_value} \Rightarrow \beta}\ \textsc{Ty\_Pval\_Loaded}$$

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Unit} \Rightarrow \texttt{unit}}\ \textsc{Ty\_Pval\_Unit}$$

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{True} \Rightarrow \texttt{bool}}\ \textsc{Ty\_Pval\_True}$$

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{False} \Rightarrow \texttt{bool}}\ \textsc{Ty\_Pval\_False}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{value}_i \Rightarrow \beta}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \beta[\,\overline{\mathit{value}_i}^{\,i}\,] \Rightarrow \texttt{list}\,\beta}\ \textsc{Ty\_Pval\_List}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{value}_i \Rightarrow \beta_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash (\,\overline{\mathit{value}_i}^{\,i}\,) \Rightarrow \overline{\beta_i}^{\,i}}\ \textsc{Ty\_Pval\_Tuple}$$

$$\frac{\texttt{smt}\,(\Phi \Rightarrow \texttt{false})}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{error}\,(\mathit{string}, \mathit{pval}) \Rightarrow \beta}\ \textsc{Ty\_Pval\_Error}$$

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Nil}\,\beta(\,) \Rightarrow \texttt{list}\,\beta}\ \textsc{Ty\_Pval\_Ctor\_Nil}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \texttt{list}\,\beta\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Cons}(\mathit{pval}_1, \mathit{pval}_2) \Rightarrow \texttt{list}\,\beta}\ \textsc{Ty\_Pval\_Ctor\_Cons}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta_i}^{\,i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{Tuple}(\overline{pval_i}^{\,i}) \Rightarrow \overline{\beta_i}^{\,i}} \quad \textsc{Ty\_Pval\_Ctor\_Tuple}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta}^{\,i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{Array}(\overline{pval_i}^{\,i}) \Rightarrow \texttt{array}\,\beta} \quad \textsc{Ty\_Pval\_Ctor\_Array}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{Specified}(pval) \Rightarrow \beta} \quad \textsc{Ty\_Pval\_Ctor\_Specified}$$

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_i \Rightarrow \beta_i}^{\,i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash (\,\texttt{struct}\,tag)\{\,\overline{.member_i = pval_i}^{\,i}\,\} \Rightarrow \texttt{struct}\,tag} \quad \textsc{Ty\_Pval\_Struct}$$

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

$$\frac{}{\Phi \vdash \texttt{emp} \;\equiv\; \texttt{emp}} \quad \textsc{TyRes\_Eq\_Emp}$$

$$\frac{\texttt{smt}\,(\Phi \Rightarrow (term_1 = term_1') \wedge (term_2 = term_2'))}{\Phi \vdash term_1\; q \overset{init}{\mapsto}_\tau term_2 \;\equiv\; term_1'\; q \overset{init}{\mapsto}_\tau term_2'} \quad \textsc{TyRes\_Eq\_PointsTo}$$

$$\frac{\Phi \vdash res_1 \;\equiv\; res_1' \qquad \Phi \vdash res_2 \;\equiv\; res_2'}{\Phi \vdash res_1 * res_2 \;\equiv\; res_1' * res_2'} \quad \textsc{TyRes\_Eq\_SepConj}$$

$$\frac{\Phi \vdash res \;\equiv\; res'}{\Phi \vdash \exists\, ident{:}\beta.\; res \;\equiv\; \exists\, ident{:}\beta.\; res'} \quad \textsc{TyRes\_Eq\_Exists}$$

$$\frac{\begin{array}{c}\texttt{smt}\,(\Phi, term \Rightarrow term') \\ \texttt{smt}\,(\Phi, term' \Rightarrow term) \\ \Phi \vdash res \;\equiv\; res'\end{array}}{\Phi \vdash term \wedge res \;\equiv\; term' \wedge res'}\ \textsc{TyRes\_Eq\_Term}$$

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

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \texttt{emp} \Leftarrow \texttt{emp}}\ \textsc{TyRes\_Emp}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \texttt{pt} \Leftarrow points\_to}\ \textsc{TyRes\_PointsTo}$$

$$\frac{\Phi \vdash res \;\equiv\; res'}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, r{:}res \vdash r \Leftarrow res'}\ \textsc{TyRes\_Var}$$

$$\frac{\begin{array}{c}\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res\_term_1 \Leftarrow res_1 \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash res\_term_2 \Leftarrow res_2\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \langle res\_term_1, res\_term_2 \rangle \Leftarrow res_1 * res_2}\ \textsc{TyRes\_SepConj}$$

$$\frac{\begin{array}{c}\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term_2 \Leftarrow [pval/y]res\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{pack}\,(pval, res\_term_2) \Leftarrow \exists\, y{:}\beta.\; res}\ \textsc{TyRes\_Pack}$$

$\boxed{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: arg \gg ret}$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash ::ret \gg ret}\ \textsc{Ty\_Spine\_Empty}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta \quad \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: [pval/x]arg \gg ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash pval,\ \overline{spine\_elem_i}^{\,i} :: \Pi\, x{:}\beta.\ arg \gg ret}\ \textsc{Ty\_Spine\_Comp}$$

$$\frac{\mathcal{L} \vdash term{:}\beta \quad \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: [term/x]arg \gg ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash term,\ \overline{spine\_elem_i}^{\,i} :: \forall\, x{:}\beta.\ arg \gg ret}\ \textsc{Ty\_Spine\_Log}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow res \quad \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_2 \vdash \overline{spine\_elem_i}^{\,i} :: arg \gg ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1,\mathcal{R}_2 \vdash res\_term,\ \overline{spine\_elem_i}^{\,i} :: res \multimap arg \gg ret}\ \textsc{Ty\_Spine\_Res}$$

$$\frac{\texttt{smt}\,(\Phi \Rightarrow term) \quad \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: arg \gg ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: term \supset arg \gg ret}\ \textsc{Ty\_Spine\_Cons}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta}{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow y{:}\beta.\ y = pval}\ \textsc{Ty\_PE\_Val}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{loc} \quad \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{integer}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{array\_shift}\,(pval_1,\tau,pval_2) \Rightarrow y{:}\texttt{loc}.\ y = pval_1 +_{\text{ptr}} (pval_2 \times \text{size\_of}(\tau))}\ \textsc{Ty\_PE\_Array\_Shift}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{loc} \quad \_{:}\,\texttt{struct}\,tag\ \&\ \overline{member_i{:}\tau_i}^{\,i} \in \texttt{Globals}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{member\_shift}\,(pval,tag,member_j) \Rightarrow y{:}\texttt{loc}.\ y = pval +_{\text{ptr}} \text{offset\_of}_{tag}(member_j)}\ \textsc{Ty\_PE\_Member\_Shift}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{bool}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{not}\,(pval) \Rightarrow y{:}\texttt{bool}.\ y = \neg\, pval}\ \textsc{Ty\_PE\_Not}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{integer} \\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{integer}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash pval_1\ \ binop_{arith}\ \ pval_2 \Rightarrow y{:}\texttt{integer}.\ y = (pval_1\ binop_{arith}\ pval_2)}\ \textsc{Ty\_PE\_Arith\_Binop}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{integer} \\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{integer}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash pval_1\ \ binop_{rel}\ \ pval_2 \Rightarrow y{:}\texttt{bool}.\ y = (pval_1\ binop_{rel}\ pval_2)}\ \textsc{Ty\_PE\_Rel\_Binop}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{bool} \\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{bool}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash pval_1\ \ binop_{bool}\ \ pval_2 \Rightarrow y{:}\texttt{bool}.\ y = (pval_1\ binop_{bool}\ pval_2)}\ \textsc{Ty\_PE\_Bool\_Binop}$$

$$\frac{\begin{array}{c}name{:}pure\_arg\ \in\ \texttt{Globals} \\ \mathcal{C};\mathcal{L};\Phi;\cdot \vdash \overline{pval_i}^{\,i}\ ::\ pure\_arg \gg \Sigma\, y'{:}\beta'.\ term' \wedge \texttt{I}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash name(\overline{pval_i}^{\,i}) \Rightarrow y'{:}\beta'.\ term'}\ \textsc{Ty\_PE\_Call}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{bool} \\ \texttt{smt}\,(\Phi \Rightarrow pval)\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{assert\_undef}\,(pval,\ UB\_name) \Rightarrow y{:}\texttt{unit}.\ y = \texttt{unit}}\ \textsc{Ty\_PE\_Assert\_Undef}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{bool}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{bool\_to\_integer}\,(pval) \Rightarrow y{:}\texttt{integer}.\ y = \texttt{if}\, pval\, \texttt{then}\, 1\, \texttt{else}\, 0}\ \textsc{Ty\_PE\_Bool\_To\_Integer}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{integer} \\ abbrev_1\ \equiv\ \mathrm{max\_int}_\tau - \mathrm{min\_int}_\tau + 1 \\ abbrev_2\ \equiv\ pval\, \texttt{rem\_f}\, abbrev_1\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{wrapI}\,(\tau, pval) \Rightarrow y{:}\beta.\ y = \texttt{if}\ abbrev_2 \leq \mathrm{max\_int}_\tau\ \texttt{then}\ abbrev_2\ \texttt{else}\ abbrev_2 - abbrev_1}\ \textsc{Ty\_PE\_WrapI}$$

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

$$\frac{}{term\ \texttt{as}\ \_{:}\beta{:}\beta \rightsquigarrow \cdot; \cdot}\ \textsc{Ty\_Pat\_Comp\_No\_Sym\_Annot}$$

$$\frac{}{term\ \texttt{as}\ x{:}\beta{:}\beta \rightsquigarrow \cdot, x{:}\beta; \cdot, x = term}\ \textsc{Ty\_Pat\_Comp\_Sym\_Annot}$$

$$\frac{}{term\ \texttt{as}\ \texttt{Nil}\ \beta(\,){:}\texttt{list}\ \beta \rightsquigarrow \cdot; \cdot}\ \textsc{Ty\_Pat\_Comp\_Nil}$$

$$\frac{\begin{array}{c} term^{(1)}\ \texttt{as}\ pattern_1{:}\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \texttt{tl}\ term\ \texttt{as}\ pattern_2{:}\texttt{list}\ \beta \rightsquigarrow \mathcal{C}_2; \Phi_1 \end{array}}{term\ \texttt{as}\ \texttt{Cons}(pattern_1, pattern_2){:}\texttt{list}\ \beta \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \Phi_1, \Phi_2}\ \textsc{Ty\_Pat\_Comp\_Cons}$$

$$\frac{\overline{term^{(i)}\ \texttt{as}\ pattern_i{:}\beta_i \rightsquigarrow \mathcal{C}_i; \Phi_i}^{\,i}}{term\ \texttt{as}\ \texttt{Tuple}(\overline{pattern_i}^{\,i}){:}\overline{\beta_i}^{\,i} \rightsquigarrow \overline{\mathcal{C}_i}^{\,i}; \overline{\Phi_i}^{\,i}}\ \textsc{Ty\_Pat\_Comp\_Tuple}$$

$$\frac{\overline{term[i]\ \texttt{as}\ pattern_i{:}\beta \rightsquigarrow \mathcal{C}_i; \Phi_i}^{\,i}}{term\ \texttt{as}\ \texttt{Array}(\overline{pattern_i}^{\,i}){:}\texttt{array}\ \beta \rightsquigarrow \overline{\mathcal{C}_i}^{\,i}; \overline{\Phi_i}^{\,i}}\ \textsc{Ty\_Pat\_Comp\_Array}$$

$$\frac{term\ \texttt{as}\ pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi}{term\ \texttt{as}\ \texttt{Specified}(pattern){:}\beta \rightsquigarrow \mathcal{C}; \Phi}\ \textsc{Ty\_Pat\_Comp\_Specified}$$

$\boxed{term\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi}$

$$\frac{}{term\ \texttt{as}\ x{:}\beta \rightsquigarrow \cdot, x{:}\beta; \cdot, x = term}\ \textsc{Ty\_Pat\_Sym\_Or\_Pattern\_Sym}$$

$$\frac{term\ \texttt{as}\ pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi}{term\ \texttt{as}\ pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi}\ \textsc{Ty\_Pat\_Sym\_Or\_Pattern\_Pattern}$$

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

$$\frac{}{\texttt{emp}{:}\texttt{emp} \rightsquigarrow \cdot; \cdot; \cdot}\ \textsc{Ty\_Pat\_Res\_Empty}$$

$$\frac{}{\texttt{pt}{:}points\_to \rightsquigarrow \cdot; \cdot; \cdot, r{:}points\_to}\ \textsc{Ty\_Pat\_Res\_PointsTo}$$

$$\frac{}{r{:}res \rightsquigarrow \cdot; \cdot; \cdot, r{:}res}\ \textsc{Ty\_Pat\_Res\_Var}$$

$$\frac{\begin{array}{c} res\_pattern_1{:}res_1 \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ res\_pattern_2{:}res_2 \rightsquigarrow \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\langle res\_pattern_1, res\_pattern_2\rangle{:}res_1 * res_2 \rightsquigarrow \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2}\ \textsc{Ty\_Pat\_Res\_SepConj}$$

$$\frac{res\_pattern{:}res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}{res\_pattern{:}term \wedge res \rightsquigarrow \mathcal{L}; \Phi, term; \mathcal{R}}\ \textsc{Ty\_Pat\_Res\_Conj}$$

$$\frac{res\_pattern{:}[x/y]res \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}{\texttt{pack}\,(x, res\_pattern){:}\exists\, y{:}\beta.\ res \rightsquigarrow \mathcal{L}, x{:}\beta; \Phi; \mathcal{R}}\ \textsc{Ty\_Pat\_Res\_Pack}$$

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

$$\frac{}{:\texttt{I} \rightsquigarrow \cdot; \cdot; \cdot; \cdot}\ \textsc{Ty\_Pat\_Ret\_Empty}$$

$$\frac{\begin{array}{c} y\ \texttt{as}\ \mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\texttt{comp}\ \mathit{ident\_or\_pattern},\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \Sigma\, y{:}\beta.\ \mathit{ret} \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2}\ \textsc{Ty\_Pat\_Ret\_Comp}$$

$$\frac{\overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\texttt{log}\ y,\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \exists\, y{:}\beta.\ \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}, y{:}\beta; \Phi; \mathcal{R}}\ \textsc{Ty\_Pat\_Ret\_Log}$$

$$\frac{\begin{array}{c} \mathit{res\_pattern}{:}\mathit{res} \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\mathit{res}\ \mathit{res\_pattern},\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{res} \otimes \mathit{ret} \rightsquigarrow \mathcal{C}_2; \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2}\ \textsc{Ty\_Pat\_Ret\_Res}$$

$$\frac{\overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{term} \wedge \mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi, \mathit{term}; \mathcal{R}}\ \textsc{Ty\_Pat\_Ret\_Cons}$$

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

$$\frac{\texttt{smt}\,(\Phi \Rightarrow \texttt{false})}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{undef}\ \mathit{UB\_name} \Leftarrow y{:}\beta.\ \mathit{term}}\ \textsc{Ty\_TPVal\_Undef}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \\ \texttt{smt}\,(\Phi \Rightarrow [\mathit{pval}/y]\mathit{term}) \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{done}\,\mathit{pval} \Leftarrow y{:}\beta.\ \mathit{term}}\ \textsc{Ty\_TPVal\_Done}$$

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

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \texttt{bool} \\ \mathcal{C}; \mathcal{L}; \Phi, \mathit{pval} = \texttt{true} \vdash \mathit{tpexpr}_1 \Leftarrow y{:}\beta.\ \mathit{term} \\ \mathcal{C}; \mathcal{L}; \Phi, \mathit{pval} = \texttt{false} \vdash \mathit{tpexpr}_2 \Leftarrow y{:}\beta.\ \mathit{term} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{if}\ \mathit{pval}\ \texttt{then}\ \mathit{tpexpr}_1\ \texttt{else}\ \mathit{tpexpr}_2 \Leftarrow y{:}\beta.\ \mathit{term}}\ \textsc{Ty\_TPE\_If}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pexpr \Rightarrow y_1{:}\beta_1.\ term_1 \\ y_1\ \texttt{as}\ ident\_or\_pattern{:}\beta_1 \rightsquigarrow \mathcal{C}_1;\Phi_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},y_1{:}\beta_1;\Phi,term_1,\Phi_1 \vdash tpexpr \Leftarrow y_2{:}\beta_2.\ term_2\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{let}\ ident\_or\_pattern = pexpr\ \texttt{in}\ tpexpr \Leftarrow y_2{:}\beta_2.\ term_2}\ \textsc{Ty\_TPE\_Let}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash tpexpr_1 \Leftarrow y_1{:}\beta_1.\ term_1 \\ y_1\ \texttt{as}\ ident\_or\_pattern{:}\beta_1 \rightsquigarrow \mathcal{C}_1;\Phi_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},y_1{:}\beta_1;\Phi,term_1,\Phi_1 \vdash tpexpr \Leftarrow y_2{:}\beta_2.\ term_2\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{let}\ ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr_1\ \texttt{in}\ tpexpr_2 \Leftarrow y_2{:}\beta_2.\ term_2}\ \textsc{Ty\_TPE\_LetT}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta_1 \\ \overline{y_1\ \texttt{as}\ pattern_i{:}\beta_1 \rightsquigarrow \mathcal{C}_i;\Phi_i}^{\,i} \\ \overline{\mathcal{C},\mathcal{C}_i;\mathcal{L},y_1{:}\beta_1;\Phi,y_1 = pval,\Phi_i \vdash tpexpr_i \Leftarrow y_2{:}\beta_2.\ term_2}^{\,i}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{case}\ pval\ \texttt{of}\ \overline{\mid\ pattern_i \Rightarrow tpexpr_i}^{\,i}\ \texttt{end} \Leftarrow y_2{:}\beta_2.\ term_2}\ \textsc{Ty\_TPE\_Case}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{integer}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{create}\,(pval,\tau) \Rightarrow \Sigma\, y_p{:}\texttt{loc}.\ \texttt{representable}\,(\tau*,y_p) \wedge \texttt{alignedI}\,(pval,y_p) \wedge \exists\, y{:}\beta_\tau.\ \langle y_p\ 1 \overset{\times}{\mapsto}_\tau y\rangle \otimes \texttt{I}}\ \textsc{Ty\_Action\_Create}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{loc} \\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \beta_\tau \\ \texttt{smt}\,(\Phi \Rightarrow \texttt{representable}\,(\tau,pval_2)) \\ \texttt{smt}\,(\Phi \Rightarrow pval_0 = pval_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,r{:}\langle pval_0\ 1 \mapsto_\tau \_\rangle \vdash \texttt{store}\,(\_,\tau,pval_1,pval_2,\_) \Rightarrow \Sigma\,\_{:}\texttt{unit}.\ pval_0\ 1 \overset{\checkmark}{\mapsto}_\tau pval_2 \otimes \texttt{I}}\ \textsc{Ty\_Action\_Store}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{loc} \\ \texttt{smt}\,(\Phi \Rightarrow pval_0 = pval_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,r{:}\langle pval_0\ 1 \overset{\checkmark}{\mapsto}_\tau pval_2\rangle \vdash \texttt{load}\,(\tau,pval_1,\_) \Rightarrow \Sigma\, y{:}\beta_\tau.\ y = pval_2 \wedge \langle pval_0\ 1 \overset{\checkmark}{\mapsto}_\tau pval_2\rangle \otimes \texttt{I}}\ \textsc{Ty\_Action\_Load}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{loc} \\ \texttt{smt}\,(\Phi \Rightarrow pval_0 = pval_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot, r{:}\langle pval_0\ 1{\mapsto_\tau}\ \_\rangle \vdash \texttt{kill}\,(\texttt{static}\,\tau, pval_1) \Rightarrow \Sigma\,\_{:}\texttt{unit}.\ \texttt{I}}\ \textsc{Ty\_Action\_Kill\_Static}$$

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ \Leftarrow \texttt{I}}\ \textsc{Ty\_TVal\_I}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow [pval/y]ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\,pval,\ \overline{spine\_elem_i}^{\,i} \Leftarrow \Sigma\,y{:}\beta.\ ret}\ \textsc{Ty\_TVal\_Comp}$$

$$\frac{\begin{array}{c}\mathcal{L} \vdash term{:}\beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow [term/y]ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\,term,\ \overline{spine\_elem_i}^{\,i} \Leftarrow \exists\,y{:}\beta.\ ret}\ \textsc{Ty\_TVal\_Log}$$

$$\frac{\begin{array}{c}\texttt{smt}\,(\Phi \Rightarrow term) \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ spine \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ spine \Leftarrow term \wedge ret}\ \textsc{Ty\_TVal\_Cons}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow res \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_2 \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1,\mathcal{R}_2 \vdash \texttt{done}\ res\_term,\ \overline{spine\_elem}^{\,i} \Leftarrow res \otimes ret}\ \textsc{Ty\_TVal\_Res}$$

$$\frac{\texttt{smt}\,(\Phi \Rightarrow \texttt{false})}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{undef}\ \ UB\_name \Leftarrow ret}\ \textsc{Ty\_TVal\_UB}$$

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \texttt{loc} \\ \mathit{pval}{:}\mathit{arg} \in \texttt{Globals} \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{ccall}\,(\tau, \mathit{pval}, \overline{\mathit{spine\_elem}_i}^{\,i}) \Rightarrow \mathit{ret}}\ \textsc{Ty\_Seq\_E\_CCall}$$

$$\frac{\begin{array}{c}\mathit{name}{:}\mathit{arg} \in \texttt{Globals} \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{pcall}\,(\mathit{name}, \overline{\mathit{spine\_elem}_i}^{\,i}) \Rightarrow \mathit{ret}}\ \textsc{Ty\_Seq\_E\_Proc}$$

$\boxed{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret}}$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \texttt{loc}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{memop}\,(\texttt{intFromPtr}\,(\tau_1, \tau_2, \mathit{pval})) \Rightarrow \Sigma\, y{:}\texttt{integer}.\ y = \texttt{cast\_ptr\_to\_int}\,\mathit{pval} \wedge \texttt{I}}\ \textsc{Ty\_Is\_E\_Memop\_IntFromPtr}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \texttt{integer}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{memop}\,(\texttt{ptrFromInt}\,(\tau_1, \tau_2, \mathit{pval})) \Rightarrow \Sigma\, y{:}\texttt{loc}.\ y = \texttt{cast\_int\_to\_ptr}\,\mathit{pval} \wedge \texttt{I}}\ \textsc{Ty\_Is\_E\_Memop\_PtrFromInt}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \texttt{loc} \\ \texttt{smt}\,(\Phi \Rightarrow \mathit{pval}_0 = \mathit{pval}_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot, r{:}\langle \mathit{pval}_0 \ \_ \overset{\checkmark}{\mapsto}_{\tau} \ \_\rangle \vdash \texttt{memop}\,(\texttt{ptrValidForDeref}\,(\tau, \mathit{pval}_1)) \Rightarrow \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, \mathit{pval}_0) \wedge \texttt{I}}\ \textsc{Ty\_Is\_E\_Memop\_PtrValidForDeref}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \texttt{loc}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{memop}\,(\texttt{ptrWellAligned}\,(\tau, \mathit{pval})) \Rightarrow \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau, \mathit{pval}) \wedge \texttt{I}}\ \textsc{Ty\_Is\_E\_Memop\_PtrWellAligned}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \texttt{loc} \\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \texttt{integer}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{memop}\,(\texttt{ptrArrayShift}\,(\mathit{pval}_1, \tau, \mathit{pval}_2)) \Rightarrow \Sigma\, y{:}\texttt{loc}.\ y = \mathit{pval}_1 +_{\text{ptr}} (\mathit{pval}_2 \times \text{size\_of}(\tau)) \wedge \texttt{I}}\ \textsc{Ty\_Is\_E\_Memop\_PtrArrayShift}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash\ mem\_action \Rightarrow ret}\ \textsc{Ty\_Is\_E\_Action}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{neg}\, mem\_action \Rightarrow ret}\ \textsc{Ty\_Is\_E\_Neg\_Action}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash tval \Leftarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash tval \Leftarrow ret}\ \textsc{Ty\_Seq\_TE\_TVal}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pexpr \Rightarrow y{:}\beta.\ term \\ y\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1;\Phi_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},y{:}\beta;\Phi,term,\Phi_1;\mathcal{R} \vdash texpr \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, texpr \Leftarrow ret}\ \textsc{Ty\_Seq\_TE\_LetP}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash tpexpr \Leftarrow y{:}\beta.\ term \\ y\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1;\Phi_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},y{:}\beta;\Phi,term,\Phi_1;\mathcal{R} \vdash texpr \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr\, \texttt{in}\, texpr \Leftarrow ret}\ \textsc{Ty\_Seq\_TE\_LetPT}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash seq\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash texpr \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}',\mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr \Leftarrow ret_2}\ \textsc{Ty\_Seq\_TE\_Let}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash texpr_1 \Leftarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash texpr_2 \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}',\mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = texpr_1\, \texttt{in}\, texpr_2 \Leftarrow ret_2}\ \textsc{Ty\_Seq\_TE\_LetT}$$

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{bool} \\ \mathcal{C};\mathcal{L};\Phi,pval = \texttt{true};\mathcal{R} \vdash texpr_1 \Leftarrow ret \\ \mathcal{C};\mathcal{L};\Phi,pval = \texttt{false};\mathcal{R} \vdash texpr_2 \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{if}\,pval\,\texttt{then}\,texpr_1\,\texttt{else}\,texpr_2 \Leftarrow ret}\ \textsc{Ty\_Seq\_TE\_If}$$

$$\frac{\begin{array}{l}ident{:}arg \in \texttt{Globals} \\ \mathcal{C};\mathcal{L};\Phi;\cdot \vdash \overline{pval_i}^{\,i} :: arg \gg \texttt{false} \wedge \texttt{I}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{run}\,ident\,\overline{pval_i}^{\,i} \Leftarrow \texttt{false} \wedge \texttt{I}}\ \textsc{Ty\_Seq\_TE\_Run}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_texpr \Leftarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{bound}\,[int](is\_texpr) \Leftarrow ret}\ \textsc{Ty\_Seq\_TE\_Bound}$$

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} {:} ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash texpr \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{let}\,\texttt{strong}\,\overline{ret\_pattern_i}^{\,i} = is\_expr\,\texttt{in}\,texpr \Leftarrow ret_2}\ \textsc{Ty\_Is\_TE\_LetS}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_texpr \Leftarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_texpr \Leftarrow ret}\ \textsc{Ty\_TE\_Is}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret}} \quad \textsc{Ty\_TE\_Seq}$$

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

$$\frac{}{\_ : \_ = \mathit{pval} \rightsquigarrow \cdot} \quad \textsc{Op\_Decons\_Value\_No\_Sym\_Annot}$$

$$\frac{}{x : \_ = \mathit{pval} \rightsquigarrow \mathit{pval}/x, \cdot} \quad \textsc{Op\_Decons\_Value\_Sym\_Annot}$$

$$\frac{\begin{array}{c}\mathit{pattern}_1 = \mathit{pval}_1 \rightsquigarrow \sigma_1 \\ \mathit{pattern}_2 = \mathit{pval}_2 \rightsquigarrow \sigma_2\end{array}}{\texttt{Cons}(\mathit{pattern}_1, \mathit{pattern}_2) = \texttt{Cons}(\mathit{pval}_1, \mathit{pval}_2) \rightsquigarrow \sigma_1, \sigma_2} \quad \textsc{Op\_Decons\_Value\_Cons}$$

$$\frac{\overline{\mathit{pattern}_i = \mathit{pval}_1 \rightsquigarrow \sigma_i}^{\,i}}{\texttt{Tuple}(\overline{\mathit{pattern}_i}^{\,i}) = \texttt{Tuple}(\overline{\mathit{pval}_i}^{\,i}) \rightsquigarrow \overline{\sigma_i}^{\,i}} \quad \textsc{Op\_Decons\_Value\_Tuple}$$

$$\frac{\overline{\mathit{pattern}_i = \mathit{pval}_1 \rightsquigarrow \sigma_i}^{\,i}}{\texttt{Array}(\overline{\mathit{pattern}_i}^{\,i}) = \texttt{Array}(\overline{\mathit{pval}_i}^{\,i}) \rightsquigarrow \overline{\sigma_i}^{\,i}} \quad \textsc{Op\_Decons\_Value\_Array}$$

$$\frac{\mathit{pattern} = \mathit{pval} \rightsquigarrow \sigma}{\texttt{Specified}(\mathit{pattern}) = \mathit{pval} \rightsquigarrow \sigma} \quad \textsc{Op\_Decons\_Value\_Specified}$$

$\boxed{\mathit{ident\_or\_pattern} = \mathit{pval} \rightsquigarrow \sigma}$

$$\frac{}{x = \mathit{pval} \rightsquigarrow \mathit{pval}/x, \cdot} \quad \textsc{Op\_Decons\_Value'\_Sym}$$

$$\frac{pattern = pval \rightsquigarrow \sigma}{pattern = pval \rightsquigarrow \sigma} \quad \textsc{Op\_Decons\_Value'\_Pattern}$$

$\boxed{res\_pattern = res\_term \rightsquigarrow \sigma}$

$$\frac{}{\texttt{emp} = \texttt{emp} \rightsquigarrow \cdot} \quad \textsc{Op\_Decons\_Res\_Emp}$$

$$\frac{}{\texttt{pt} = \texttt{pt} \rightsquigarrow \cdot} \quad \textsc{Op\_Decons\_Res\_Points\_to}$$

$$\frac{}{ident = res\_term \rightsquigarrow res\_term/ident, \cdot} \quad \textsc{Op\_Decons\_Res\_Var}$$

$$\frac{\begin{array}{c} res\_pattern_1 = res\_term_1 \rightsquigarrow \sigma_1 \\ res\_pattern_2 = res\_term_2 \rightsquigarrow \sigma_2 \end{array}}{\langle res\_pattern_1, res\_pattern_2 \rangle = \langle res\_term_1, res\_term_2 \rangle \rightsquigarrow \sigma_1, \sigma_2} \quad \textsc{Op\_Decons\_Res\_Pair}$$

$$\frac{res\_pattern = res\_term \rightsquigarrow \sigma}{\texttt{pack}\,(ident, res\_pattern) = \texttt{pack}\,(pval, res\_term) \rightsquigarrow pval/ident, \sigma} \quad \textsc{Op\_Decons\_Res\_Pack}$$

$\boxed{ret\_pattern = spine\_elem \rightsquigarrow \sigma}$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\texttt{comp}\, ident\_or\_pattern = pval \rightsquigarrow \sigma} \quad \textsc{Op\_Decons\_Ret\_Comp}$$

$$\frac{}{\texttt{log}\, ident = term \rightsquigarrow term/ident, \cdot} \quad \textsc{Op\_Decons\_Ret\_Log}$$

$$\frac{res\_pattern = res\_term \rightsquigarrow \sigma}{res\; res\_pattern = res\_term \rightsquigarrow \sigma} \quad \textsc{Op\_Decons\_Ret\_Res}$$

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

$$\frac{mem\_ptr' \equiv mem\_ptr +_{\text{ptr}} mem\_int \times \text{size\_of}(\tau)}{\langle \texttt{array\_shift}\,(mem\_ptr, \tau, mem\_int) \rangle \longrightarrow \langle mem\_ptr' \rangle} \quad \textsc{Op\_PE\_PE\_ArrayShift}$$

$$\frac{mem\_ptr' \equiv mem\_ptr +_{\text{ptr}} \text{offset\_of}_{tag}(member)}{\langle \texttt{member\_shift}\,(mem\_ptr, tag, member) \rangle \longrightarrow \langle mem\_ptr' \rangle} \quad \textsc{Op\_PE\_PE\_MemberShift}$$

$$\frac{}{\langle \texttt{not}\,(\texttt{True}) \rangle \longrightarrow \langle \texttt{False} \rangle} \quad \textsc{Op\_PE\_PE\_NotTrue}$$

$$\frac{}{\langle \texttt{not}\,(\texttt{False}) \rangle \longrightarrow \langle \texttt{True} \rangle} \quad \textsc{Op\_PE\_PE\_NotFalse}$$

$$\frac{mem\_int \equiv mem\_int_1\ binop_{arith}\ mem\_int_2}{\langle mem\_int_1\ binop_{arith}\ mem\_int_2 \rangle \longrightarrow \langle mem\_int \rangle} \quad \textsc{Op\_PE\_PE\_Arith\_Binop}$$

$$\frac{bool\_value \equiv mem\_int_1\ binop_{rel}\ mem\_int_2}{\langle mem\_int_1\ binop_{rel}\ mem\_int_2 \rangle \longrightarrow \langle bool\_value \rangle} \quad \textsc{Op\_PE\_PE\_Rel\_Binop}$$

$$\frac{bool\_value \equiv bool\_value_1\ binop_{bool}\ bool\_value_2}{\langle bool\_value_1\ binop_{bool}\ bool\_value_2 \rangle \longrightarrow \langle bool\_value \rangle} \quad \textsc{Op\_PE\_PE\_Bool\_Binop}$$

$$\frac{}{\langle \texttt{assert\_undef}\,(\texttt{True},\ UB\_name) \rangle \longrightarrow \langle \texttt{Unit} \rangle} \quad \textsc{Op\_PE\_PE\_Assert\_Undef}$$

$$\frac{}{\langle \texttt{bool\_to\_integer}\,(\texttt{True}) \rangle \longrightarrow \langle 1 \rangle} \quad \textsc{Op\_PE\_PE\_Bool\_To\_Integer\_True}$$

$$\frac{}{\langle \texttt{bool\_to\_integer}\,(\texttt{False}) \rangle \longrightarrow \langle 0 \rangle} \quad \textsc{Op\_PE\_PE\_Bool\_To\_Integer\_False}$$

$$\frac{\begin{array}{l} abbrev_1 \equiv \max\_int_\tau - \min\_int_\tau + 1 \\ abbrev_2 \equiv pval\ \texttt{rem\_f}\ abbrev_1 \\ mem\_int' \equiv \texttt{if}\ abbrev_2 \leq \max\_int_\tau\ \texttt{then}\ abbrev_2\ \texttt{else}\ abbrev_2 - abbrev_1 \end{array}}{\langle \texttt{wrapI}\,(\tau, mem\_int)\rangle \longrightarrow \langle mem\_int'\rangle}\ \textsc{Op\_PE\_PE\_WrapI}$$

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

$$\frac{name \equiv \overline{x_i}^{\,i} \mapsto tpexpr{:}(y{:}\beta.\ term) \in \texttt{Globals}}{\langle name(\overline{pval_i}^{\,i})\rangle \longrightarrow \langle \overline{pval_i/x_i, \cdot}^{\,i}(tpexpr){:}(y{:}\beta.\ \overline{pval_i/x_i, \cdot}^{\,i}(term))\rangle}\ \textsc{Op\_PE\_TPE\_Call}$$

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

$$\frac{\begin{array}{c} pattern_j = pval \rightsquigarrow \sigma_j \\ \forall\, i < j.\ \texttt{not}\,(pattern_i = pval \rightsquigarrow \sigma_i) \end{array}}{\langle \texttt{case}\, pval\, \texttt{of}\, \overline{\mid pattern_i \Rightarrow tpexpr_i}^{\,i}\, \texttt{end}\rangle \longrightarrow \langle \sigma_j(tpexpr_j)\rangle}\ \textsc{Op\_TPE\_TPE\_Case}$$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle \texttt{let}\, ident\_or\_pattern = pval\, \texttt{in}\, tpexpr\rangle \longrightarrow \langle \sigma(tpexpr)\rangle}\ \textsc{Op\_TPE\_TPE\_Let\_Sub}$$

$$\frac{\langle pexpr\rangle \longrightarrow \langle pexpr'\rangle}{\langle \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr\rangle \longrightarrow \langle \texttt{let}\, ident\_or\_pattern = pexpr'\, \texttt{in}\, tpexpr\rangle}\ \textsc{Op\_TPE\_TPE\_Let\_Let}$$

$$\frac{\langle pexpr\rangle \longrightarrow \langle tpexpr_1{:}(y{:}\beta.\ term)\rangle}{\langle \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr_2\rangle \longrightarrow \langle \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr_1\, \texttt{in}\, tpexpr_2\rangle}\ \textsc{Op\_TPE\_TPE\_Let\_LetT}$$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = \texttt{done}\, pval\, \texttt{in}\, tpexpr\rangle \longrightarrow \langle \sigma(tpexpr)\rangle}\ \textsc{Op\_TPE\_TPE\_LetT\_Sub}$$

$$\frac{\langle tpexpr_1\rangle \longrightarrow \langle tpexpr_1'\rangle}{\langle \texttt{let}\ ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr_1\ \texttt{in}\ tpexpr_2\rangle \longrightarrow \langle \texttt{let}\ ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr_1'\ \texttt{in}\ tpexpr_2\rangle}\ \textsc{Op\_TPE\_TPE\_LetT\_LetT}$$

$$\frac{}{\langle \texttt{if True then}\ tpexpr_1\ \texttt{else}\ tpexpr_2\rangle \longrightarrow \langle tpexpr_1\rangle}\ \textsc{Op\_TPE\_TPE\_If\_True}$$

$$\frac{}{\langle \texttt{if False then}\ tpexpr_1\ \texttt{else}\ tpexpr_2\rangle \longrightarrow \langle tpexpr_2\rangle}\ \textsc{Op\_TPE\_TPE\_If\_False}$$

$\boxed{\langle seq\_expr\rangle \longrightarrow \langle texpr{:}ret\rangle}$

$$\frac{pval \equiv \overline{x_i}^{\,i} \mapsto texpr{:}ret \in \texttt{Globals}}{\langle \texttt{ccall}\ (\tau, pval, \overline{spine\_elem_i}^{\,i})\rangle \longrightarrow \langle \overline{spine\_elem_i/x_i, \cdot}^{\,i}(texpr){:}\overline{spine\_elem_i/x_i, \cdot}^{\,i}(ret)\rangle}\ \textsc{Op\_SE\_TE\_CCall}$$

$$\frac{name \equiv \overline{x_i}^{\,i} \mapsto texpr{:}ret \in \texttt{Globals}}{\langle \texttt{pcall}\ (name, \overline{spine\_elem_i}^{\,i})\rangle \longrightarrow \langle \overline{spine\_elem_i/x_i, \cdot}^{\,i}(texpr){:}\overline{spine\_elem_i/x_i, \cdot}^{\,i}(ret)\rangle}\ \textsc{Op\_SE\_TE\_PCall}$$

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

$$\frac{ident \equiv \overline{x_i}^{\,i} \mapsto texpr{:}\_ \in \texttt{Globals}}{\langle h; \texttt{run}\ ident\ \overline{pval_i}^{\,i}\rangle \longrightarrow \langle h; \overline{pval_i/x_i, \cdot}^{\,i}(texpr)\rangle}\ \textsc{Op\_STE\_TE\_Run}$$

$$\frac{\begin{array}{c} pattern_j = pval \rightsquigarrow \sigma_j \\ \forall\, i < j.\ \texttt{not}\ (pattern_i = pval \rightsquigarrow \sigma_i) \end{array}}{\langle h; \texttt{case}\ pval\ \texttt{of}\ |\ \overline{pattern_i \Rightarrow texpr_i}^{\,i}\ \texttt{end}\rangle \longrightarrow \langle h; \sigma_j(texpr_j)\rangle}\ \textsc{Op\_STE\_TE\_Case}$$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle h; \texttt{let}\ ident\_or\_pattern = pval\ \texttt{in}\ texpr\rangle \longrightarrow \langle h; \sigma(texpr)\rangle}\ \textsc{Op\_STE\_TE\_LetP\_Sub}$$

$$\frac{\langle pexpr\rangle \longrightarrow \langle pexpr'\rangle}{\langle h; \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, texpr\rangle \longrightarrow \langle h; \texttt{let}\, ident\_or\_pattern = pexpr'\, \texttt{in}\, texpr\rangle}\ \textsc{Op\_STE\_TE\_LetP\_LetP}$$

$$\frac{\langle pexpr\rangle \longrightarrow \langle tpexpr{:}(y{:}\beta.\ term)\rangle}{\langle h; \texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, texpr\rangle \longrightarrow \langle h; \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr\, \texttt{in}\, texpr\rangle}\ \textsc{Op\_STE\_TE\_LetP\_LetTP}$$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle h; \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = \texttt{done}\, pval\, \texttt{in}\, texpr\rangle \longrightarrow \langle h; \sigma(texpr)\rangle}\ \textsc{Op\_STE\_TE\_LetTP\_Sub}$$

$$\frac{\langle tpexpr\rangle \longrightarrow \langle tpexpr'\rangle}{\langle h; \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr\, \texttt{in}\, texpr\rangle \longrightarrow \langle h; \texttt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr'\, \texttt{in}\, texpr\rangle}\ \textsc{Op\_STE\_TE\_LetTP\_LetTP}$$

$$\frac{\overline{ret\_pattern_i = spine\_elem_i \rightsquigarrow \sigma_i}^{\,i}}{\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} {:} ret = \texttt{done}\, \overline{spine\_elem_i}^{\,i}\, \texttt{in}\, texpr\rangle \longrightarrow \langle h; \overline{\sigma_i}^{\,i}(texpr)\rangle}\ \textsc{Op\_STE\_TE\_LetT\_Sub}$$

$$\frac{\langle seq\_expr\rangle \longrightarrow \langle texpr_1{:}ret\rangle}{\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr_2\rangle \longrightarrow \langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} {:} ret = texpr_1\, \texttt{in}\, texpr_2\rangle}\ \textsc{Op\_STE\_TE\_Let\_LetT}$$

$$\frac{\langle h; texpr_1\rangle \longrightarrow \langle h'; texpr_1'\rangle}{\langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} {:} ret = texpr_1\, \texttt{in}\, texpr_2\rangle \longrightarrow \langle h; \texttt{let}\, \overline{ret\_pattern_i}^{\,i} {:} ret = texpr_1'\, \texttt{in}\, texpr_2\rangle}\ \textsc{Op\_STE\_TE\_LetT\_LetT}$$

$$\frac{}{\langle h; \texttt{if True then}\, texpr_1\, \texttt{else}\, texpr_2\rangle \longrightarrow \langle h; texpr_1\rangle}\ \textsc{Op\_STE\_TE\_If\_True}$$

$$\frac{}{\langle h; \texttt{if False then}\, texpr_1\, \texttt{else}\, texpr_2\rangle \longrightarrow \langle h; texpr_2\rangle}\ \textsc{Op\_STE\_TE\_If\_False}$$

$$\frac{}{\langle h; \texttt{bound}\, [int](is\_texpr)\rangle \longrightarrow \langle h; is\_texpr\rangle}\ \textsc{Op\_STE\_TE\_Bound}$$

$\boxed{\langle mem\_op \rangle \longrightarrow \langle pval \rangle}$

$$\frac{mem\_int \equiv \texttt{cast\_ptr\_to\_int}\, pval}{\langle \texttt{intFromPtr}\, (\tau_1, \tau_2, pval) \rangle \longrightarrow \langle mem\_int \rangle} \quad \textsc{Op\_Memop\_Pval\_IntFromPtr}$$

$$\frac{mem\_ptr \equiv \texttt{cast\_ptr\_to\_int}\, pval}{\langle \texttt{ptrFromInt}\, (\tau_1, \tau_2, pval) \rangle \longrightarrow \langle mem\_ptr \rangle} \quad \textsc{Op\_Memop\_Pval\_PtrFromInt}$$

$$\frac{bool\_value \equiv \texttt{aligned}\, (\tau, pval)}{\langle \texttt{ptrValidForDeref}\, (\tau, pval) \rangle \longrightarrow \langle bool\_value \rangle} \quad \textsc{Op\_Memop\_Pval\_PtrValidForDeref}$$

$$\frac{bool\_value \equiv \texttt{aligned}\, (\tau, pval)}{\langle \texttt{ptrWellAligned}\, (\tau, pval) \rangle \longrightarrow \langle bool\_value \rangle} \quad \textsc{Op\_Memop\_Pval\_PtrWellAligned}$$

$$\frac{mem\_ptr \equiv pval_1 +_{\text{ptr}} (pval_2 \times \text{size\_of}(\tau))}{\langle \texttt{ptrArrayShift}\, (pval_1, \tau, pval_2) \rangle \longrightarrow \langle mem\_ptr \rangle} \quad \textsc{Op\_Memop\_Pval\_PtrArrayShift}$$

$\boxed{\langle h; mem\_action \rangle \longrightarrow \langle h'; spine \rangle}$

$$\frac{\texttt{fresh}\, (mem\_ptr)}{\langle h; \texttt{create}\, (pval, \tau) \rangle \longrightarrow \langle h + \{mem\_ptr \mapsto \times\}; mem\_ptr, \_, \texttt{pt} \rangle} \quad \textsc{Op\_Action\_Spine\_Create}$$

$$\frac{}{\langle h + \{mem\_ptr \mapsto \_\}; \texttt{store}\, (\_, \tau, mem\_ptr, pval, \_) \rangle \longrightarrow \langle h + \{mem\_ptr \mapsto pval\}; \texttt{Unit}, \texttt{pt} \rangle} \quad \textsc{Op\_Action\_Spine\_Store}$$

$$\frac{}{\langle h + \{mem\_ptr \mapsto \_\}; \texttt{kill}\, (\texttt{static}\, \tau, mem\_ptr) \rangle \longrightarrow \langle h; \texttt{Unit} \rangle} \quad \textsc{Op\_Action\_Spine\_Kill\_Static}$$

$\boxed{\langle h; is\_expr \rangle \longrightarrow \langle h'; is\_expr' \rangle}$

$$\frac{\langle mem\_op\rangle \longrightarrow \langle pval\rangle}{\langle h; \texttt{memop}\,(mem\_op)\rangle \longrightarrow \langle h; \texttt{done}\,pval\rangle}\quad \textsc{Op\_IsE\_IsE\_Memop}$$

$$\frac{\langle h; mem\_action\rangle \longrightarrow \langle h; spine\rangle}{\langle h;\ mem\_action\rangle \longrightarrow \langle h; \texttt{done}\ spine\rangle}\quad \textsc{Op\_IsE\_IsE\_Action}$$

$$\frac{\langle h; mem\_action\rangle \longrightarrow \langle h'; spine\rangle}{\langle h; \texttt{neg}\,mem\_action\rangle \longrightarrow \langle h'; \texttt{done}\ spine\rangle}\quad \textsc{Op\_IsE\_IsE\_Neg\_Action}$$

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

$$\frac{\overline{ret\_pattern_i = spine\_elem_i \rightsquigarrow \sigma_i}^{\,i}}{\langle h; \texttt{let strong}\,\overline{ret\_pattern_i}^{\,i} = \texttt{done}\,\overline{spine\_elem_i}^{\,i}\ \texttt{in}\,texpr\rangle \longrightarrow \langle h; \overline{\sigma_i}^{\,i}(texpr)\rangle}\quad \textsc{Op\_IsTE\_IsTE\_LetS\_Sub}$$

$$\frac{\langle h; is\_expr\rangle \longrightarrow \langle h'; is\_expr'\rangle}{\langle h; \texttt{let strong}\,\overline{ret\_pattern_i}^{\,i} = is\_expr\ \texttt{in}\,texpr\rangle \longrightarrow \langle h'; \texttt{let strong}\,\overline{ret\_pattern_i}^{\,i} = is\_expr'\ \texttt{in}\,texpr\rangle}\quad \textsc{Op\_IsTE\_IsTE\_LetS\_LetS}$$

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

$$\frac{\langle h; seq\_texpr\rangle \longrightarrow \langle h; texpr\rangle}{\langle h; seq\_texpr\rangle \longrightarrow \langle h; texpr\rangle}\quad \textsc{Op\_TE\_TE\_Seq}$$

$$\frac{\langle h; is\_texpr\rangle \longrightarrow \langle h'; texpr\rangle}{\langle h; is\_texpr\rangle \longrightarrow \langle h'; texpr\rangle}\quad \textsc{Op\_TE\_TE\_Is}$$

```
Definition rules:        168 good    0 bad
Definition rule clauses: 366 good    0 bad
```