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 |
| | \| | $\tau *$ | 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
	|	$\beta \to \beta'$	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
	|	/\	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
| $\mathtt{array}\,(\,\overline{loaded\_value_i}^{\,i}\,)$ C array value
| $(\,\mathtt{struct}\,ident\,)\{\,\overline{.\,member_i{:}\tau_i = mem\_val_i}^{\,i}\,\}$ C struct value
| $(\,\mathtt{union}\,ident\,)\{\,.\,member = mem\_val\,\}$ C union value

loaded_value   ::=   potentially unspecified C object values
| $\mathtt{specified}\,object\_value$ 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
| $\mathtt{Nil}\,\beta$ empty list
| Cons   list cons
| Tuple   tuple

4

	|	`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
	|	$\mathbf{constrained}\,(\overline{mem\_iv\_c_i, pval_i}^{\,i})$	constrained value
	|	$\mathbf{error}\,(string, pval)$	impl-defined static error
	|	$ctor\_val(\overline{pval_i}^{\,i})$	data constructor application
	|	$(\,\mathbf{struct}\,ident)\{\,\overline{.\,member_i = pval_i}^{\,i}\,\}$	C struct expression
	|	$(\,\mathbf{union}\,ident)\{\,.\,member = pval\}$	C union expression
tpval	::=		top-level pure values

| | | **undef** *UB_name* | | undefined behaviour |
| | | **done** *pval* | | pure done |

ident_opt_β ::=   type annotated optional identifier
| | _:β | 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

 | $\texttt{if}\ term_1\ \texttt{then}\ term_2\ \texttt{else}\ term_3$

$arith\_op$ ::=

 | $term_1 + term_2$

 | $term_1 - term_2$

 | $term_1 \times term_2$

 | $term_1 / term_2$

 | $term_1\ \texttt{rem\_t}\ term_2$

 | $term_1\ \texttt{rem\_f}\ term_2$

 | $term_1\ \texttt{\textasciicircum}\ term_2$

 | $term_1\, binop_{arith}\, term_2$ 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

$list\_op$ ::=

 | $\texttt{nil}$

 | $term_1 :: term_2$

$$\mid \quad \text{tl } term$$
$$\mid \quad term^{(int)}$$

$tuple\_op$ $::=$

$$\mid \quad (\,\overline{term_i}^{\,i}\,)$$
$$\mid \quad term^{(int)}$$

$pointer\_op$ $::=$

$$\mid \quad mem\_ptr$$
$$\mid \quad term_1 +_{\text{ptr}} term_2$$
$$\mid \quad \text{cast\_int\_to\_ptr } term$$
$$\mid \quad \text{cast\_ptr\_to\_int } term$$

$array\_op$ $::=$

$$\mid \quad [|\ \overline{term_i}^{\,i}\,|]$$
$$\mid \quad term_1[term_2]$$

$param\_op$ $::=$

$$\mid \quad ident{:}\beta.\ term$$
$$\mid \quad term(term_1, .., term_n)$$

$struct\_op$ $::=$

$$\mid \quad term.member$$

ct_pred $::=$

$$\mid \quad \text{representable}\,(\tau, term)$$
$$\mid \quad \text{aligned}\,(\tau, term)$$
$$\mid \quad \text{alignedI}\,(term_1, term_2)$$

$term,\ \_$ $::=$

	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
	$\texttt{array\_shift}\,(pval_1, \tau, pval_2)$		pointer array shift
	$\texttt{member\_shift}\,(pval, ident, member)$		pointer struct/union member shift
	$\texttt{not}\,(pval)$		boolean not
	$pval_1\ binop\ pval_2$		binary operations
	$\texttt{memberof}\,(ident, member, pval)$		C struct/union member access
	$name(\overline{pval_i}^{\,i})$		pure function call
	$\texttt{assert\_undef}\,(pval,\ UB\_name)$		
	$\texttt{bool\_to\_integer}\,(pval)$		
	$\texttt{conv\_int}\,(\tau, pval)$		
	$\texttt{wrapI}\,(\tau, pval)$		

$tpexpr$	::=		top-level pure expressions

| *tpval*   top-level pure values
| `case` *pval* `of` $\overline{\text{| } tpexpr\_case\_branch_i}^{\ i}$ `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:β_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
| σ(*tpexpr*)   M   simul-sub σ in *tpexpr*

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

res_term   ::=   resource terms
| `emp`   empty heap
| *points_to*   single-cell heap
| *ident*   variable
| $\langle res\_term_1, res\_term_2 \rangle$ seperating-conjunction pair
| `pack`(*pval*, *res_term*)   packing for existentials
| σ(*res_term*)   M   substitution for resource te

mem_action	::=		memory actions
	|	$\texttt{create}\,(pval, \tau)$	
	|	$\texttt{create\_readonly}\,(pval_1, \tau, pval_2)$	
	|	$\texttt{alloc}\,(pval_1, pval_2)$	
	|	$\texttt{kill}\,(m\_kill\_kind, pval, pt)$	
	|	$\texttt{store}\,(bool, \tau, pval_1, pval_2, mem\_order, pt)$	true means store is locking
	|	$\texttt{load}\,(\tau, pval, mem\_order, pt)$	
	|	$\texttt{rmw}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	|	$\texttt{fence}\,(mem\_order)$	
	|	$\texttt{cmp\_exch\_strong}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	|	$\texttt{cmp\_exch\_weak}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	|	$\texttt{linux\_fence}\,(linux\_mem\_order)$	
	|	$\texttt{linux\_load}\,(\tau, pval, linux\_mem\_order)$	
	|	$\texttt{linux\_store}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	
	|	$\texttt{linux\_rmw}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	

polarity	::=		polarities for memory actions
	|		(pos) sequenced by `let weak` and `let strong`
	|	$\texttt{neg}$	only sequenced by `let strong`

| *pol_mem_action* | ::= | | memory actions with polarity |
| | \| | *polarity mem_action* | |

mem_op	::=		operations involving the memory state
	|	$pval_1\ binop_{rel}\ pval_2$	pointer relational binary operations
	|	$pval_1 -_\tau pval_2$	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
	|	$\texttt{ptrValidForDeref}\,(\tau, pval, pt)$	dereferencing validity predicate
	|	$\texttt{ptrWellAligned}\,(\tau, pval)$	

	$\mathtt{ptrArrayShift}\,(pval_1, \tau, pval_2)$
	$\mathtt{memcpy}\,(pval_1, pval_2, pval_3)$
	$\mathtt{memcmp}\,(pval_1, pval_2, pval_3)$
	$\mathtt{realloc}\,(pval_1, pval_2, pval_3)$
	$\mathtt{va\_start}\,(pval_1, pval_2)$
	$\mathtt{va\_copy}\,(pval)$
	$\mathtt{va\_arg}\,(pval, \tau)$
	$\mathtt{va\_end}\,(pval)$

$spine\_elem$	$::=$		spine element
	$pval$		pure or logical value
	res_term		resource value
	$\sigma(spine\_elem)$	M	substitution for spine elements / return values

| $spine$ | $::=$ | | spine |
| | $\overline{spine\_elem_i}^{\,i}$ | | |

$tval$	$::=$		(effectful) top-level values
	$\mathtt{done}\ spine$		end of top-level expression
	$\mathtt{undef}\ UB\_name$		undefined behaviour

$res\_pattern$	$::=$		resource terms
	$\mathtt{emp}$	$\text{binders} = \{\}$	empty heap
	pt	$\text{binders} = \{\}$	single-cell heap
	$ident$	$\text{binders} = ident$	variable
	$\langle res\_pattern_1, res\_pattern_2 \rangle$	$\text{binders} = \text{binders}(res\_pattern_1) \cup \text{binders}(res\_pattern_2)$	seperating-conjunction pair
	$\mathtt{pack}\,(ident, res\_pattern)$	$\text{binders} = ident \cup \text{binders}(res\_pattern)$	packing for existentials

| $ret\_pattern$ | $::=$ | | return pattern |
| | $\mathtt{comp}\ ident\_or\_pattern$ | $\text{binders} = \text{binders}(ident\_or\_pattern)$ | computational variable |

| | $\vert$ | $\texttt{log}\,ident$ | $binders = ident$ | logical variable |
| | $\vert$ | $\texttt{res}\,res\_pattern$ | $binders = \text{binders}(res\_pattern)$ | resource variable |

$init,$	$::=$			initialisation status
	$\vert$	$\checkmark$		initialised
	$\vert$	$\times$		uninitalised

| $points\_to,\ pt$ | $::=$ | | | points-to separation logic predicate |
| | $\vert$ | $term_1 \overset{init}{\mapsto}_\tau term_2$ | | |

res	$::=$			resources
	$\vert$	$\texttt{emp}$		empty heap
	$\vert$	$points\_to$		points-top heap pred.
	$\vert$	$res_1 * res_2$		seperating conjunction
	$\vert$	$\exists\,ident{:}\beta.\ res$		existential
	$\vert$	$term \wedge res$		logical conjunction
	$\vert$	$\sigma(res)$	M	simul-sub σ in res

$ret,\ \_$	$::=$			return types
	$\vert$	$\Sigma\,ident{:}\beta.\ ret$		return a computational value
	$\vert$	$\exists\,ident{:}\beta.\ ret$		return a logical value
	$\vert$	$res \otimes ret$		return a resource value
	$\vert$	$term \wedge ret$		return a predicate (post-condition)
	$\vert$	$\texttt{I}$		end return list
	$\vert$	$\sigma(ret)$	M	simul-sub σ in ret

seq_expr	$::=$			sequential (effectful) expressions
	$\vert$	$\texttt{ccall}\,(\tau, pval, spine)$		C function call
	$\vert$	$\texttt{pcall}\,(name, spine)$		procedure call

$$
\begin{array}{llll}
\mathit{seq\_texpr} & ::= & & \text{sequential top-level (effectful) expres}\\
& | & \mathit{tval} & \text{(effectful) top-level values}\\
& | & \texttt{run}\,\mathit{ident}\,\overline{\mathit{pval}_i}^{\,i} & \text{run from label}\\
& | & \texttt{let}\,\mathit{ident\_or\_pattern} = \mathit{pexpr}\,\texttt{in}\,\mathit{texpr} & \text{bind binders}(\mathit{ident\_or\_pattern})\,\text{in}\,\mathit{texpr} & \text{pure let}\\
& | & \texttt{let}\,\mathit{ident\_or\_pattern}{:}(y_1{:}\beta_1.\,\mathit{term}_1) = \mathit{tpexpr}\,\texttt{in}\,\mathit{texpr} & \text{bind binders}(\mathit{ident\_or\_pattern})\,\text{in}\,\mathit{texpr} & \text{pure let}\\
& & & \text{bind}\,y_1\,\text{in}\,\mathit{term}_1\\
& | & \texttt{let}\,\overline{\mathit{ret\_pattern}_i}^{\,i} = \mathit{seq\_expr}\,\texttt{in}\,\mathit{texpr} & \text{bind binders}(\overline{\mathit{ret\_pattern}_i}^{\,i})\,\text{in}\,\mathit{texpr} & \text{bind return patterns}\\
& | & \texttt{let}\,\overline{\mathit{ret\_pattern}_i}^{\,i}{:}\mathit{ret} = \mathit{texpr}_1\,\texttt{in}\,\mathit{texpr}_2 & \text{bind binders}(\overline{\mathit{ret\_pattern}_i}^{\,i})\,\text{in}\,\mathit{texpr}_2 & \text{annotated bind return patterns}\\
& | & \texttt{case}\,\mathit{pval}\,\texttt{of}\,\overline{|\ \mathit{texpr\_case\_branch}_i}^{\,i}\,\texttt{end} & & \text{pattern matching}\\
& | & \texttt{if}\,\mathit{pval}\,\texttt{then}\,\mathit{texpr}_1\,\texttt{else}\,\mathit{texpr}_2 & & \text{conditional}\\
& | & \texttt{bound}\,[\mathit{int}](\mathit{is\_texpr}) & & \text{limit scope of indet seq behaviour}\\
\\
\mathit{texpr\_case\_branch} & ::= & & \text{top-level case expression branch}\\
& | & \mathit{pattern} \Rightarrow \mathit{texpr} & \text{bind binders}(\mathit{pattern})\,\text{in}\,\mathit{texpr} & \text{top-level case expression branch}\\
\\
\mathit{is\_expr} & ::= & & \text{indet seq (effectful) expressions}\\
& | & \mathit{tval} & & \text{(effectful) top-level values}\\
& | & \texttt{memop}\,(\mathit{mem\_op}) & & \text{pointer op involving memory}\\
& | & \mathit{pol\_mem\_action} & & \text{memory action}\\
\\
\mathit{is\_texpr} & ::= & & \text{indet seq top-level (effectful) express}\\
& | & \texttt{let}\,\texttt{weak}\,\overline{\mathit{ret\_pattern}_i}^{\,i} = \mathit{is\_expr}\,\texttt{in}\,\mathit{texpr} & \text{bind binders}(\overline{\mathit{ret\_pattern}_i}^{\,i})\,\text{in}\,\mathit{texpr} & \text{weak sequencing}\\
& | & \texttt{let}\,\texttt{strong}\,\overline{\mathit{ret\_pattern}_i}^{\,i} = \mathit{is\_expr}\,\texttt{in}\,\mathit{texpr} & \text{bind binders}(\overline{\mathit{ret\_pattern}_i}^{\,i})\,\text{in}\,\mathit{texpr} & \text{strong sequencing}\\
\\
\mathit{texpr} & ::= & & \text{top-level (effectful) expressions}\\
& | & \mathit{seq\_texpr} & & \text{sequential (effectful) expressions}\\
& | & \mathit{is\_texpr} & & \text{indet seq (effectful) expressions}\\
& | & \sigma(\mathit{texpr}) & \text{M} & \text{simul-sub}\,\sigma\,\text{in}\,\mathit{texpr}\\
\\
\mathit{arg} & ::= & & \text{argument/function types}\\
\end{array}
$$

| $\Pi\,ident{:}\beta.\ arg$
| $\forall\,ident{:}\beta.\ arg$
| $res \multimap arg$
| $term \supset arg$
| 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
| $\Sigma\,ident{:}\beta.\ pure\_ret$
| $term \wedge pure\_ret$
| 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
| .
| $\Phi, term$

$$\mid \quad \overline{\Phi_i}^{\,i}$$

$\mathcal{R}$ ::=   resources env

$\qquad\qquad\qquad\quad \mid \quad .$

$\qquad\qquad\qquad\quad \mid \quad \mathcal{R}, res$

$\qquad\qquad\qquad\quad \mid \quad \mathcal{R}, ident{:}res$

$\qquad\qquad\qquad\quad \mid \quad \overline{\mathcal{R}_i}^{\,i}$

$\sigma,\ \psi$ ::=   substitutions

$\qquad\qquad\qquad\quad \mid \quad .$

$\qquad\qquad\qquad\quad \mid \quad spine\_elem/ident, \sigma$

$\qquad\qquad\qquad\quad \mid \quad term/ident, \sigma$

$\qquad\qquad\qquad\quad \mid \quad \overline{\sigma_i}^{\,i}$

$\qquad\qquad\qquad\quad \mid \quad \sigma(\psi)$ M   apply σ to all elements in ψ

typing   ::=

$\qquad\qquad\qquad\quad \mid \quad \mathtt{smt}\,(\Phi \Rightarrow term)$

$\qquad\qquad\qquad\quad \mid \quad ident{:}\beta \in \mathcal{C}$

$\qquad\qquad\qquad\quad \mid \quad ident{:}\beta \in \mathcal{L}$

$\qquad\qquad\qquad\quad \mid \quad ident{:}\,\mathtt{struct}\,tag\,\&\,\overline{member_i{:}\tau_i}^{\,i} \in \mathtt{Globals}$

$\qquad\qquad\qquad\quad \mid \quad \overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash mem\_val_i \Rightarrow \mathtt{mem}\,\beta_i}^{\,i}$ dependent on memory object model

opsem   ::=

$\qquad\qquad\qquad\quad \mid \quad \forall\, i < j.\ \mathtt{not}\,(pattern_i = pval \rightsquigarrow \sigma_i)$

$\qquad\qquad\qquad\quad \mid \quad \mathtt{fresh}\,(mem\_ptr)$

$\qquad\qquad\qquad\quad \mid \quad term$

$\qquad\qquad\qquad\quad \mid \quad pval{:}\beta$

formula   ::=

$\qquad\qquad\qquad\quad \mid \quad judgement$

$$| \quad typing$$
$$| \quad opsem$$
$$| \quad term \equiv term'$$
$$| \quad name{:}pure\_arg \equiv \overline{x_i}^{\,i} \mapsto tpexpr \in \texttt{Globals}$$
$$| \quad pval{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \texttt{Globals}$$

$heap, \ h$ ::=   heaps

$$| \quad .$$
$$| \quad h + \{points\_to\}$$

$object\_value\_jtype$ ::=

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

$pval\_jtype$ ::=

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

res_jtype ::=

$$| \quad \Phi \vdash res \equiv res'$$
$$| \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Leftarrow res$$

$spine\_jtype$ ::=

$$| \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret$$

$pexpr\_jtype$ ::=

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

$comp\_pattern\_jtype$ ::=

$$| \quad pattern{:}\beta \rightsquigarrow \mathcal{C}\,\texttt{with}\,term$$
$$| \quad ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}\,\texttt{with}\,term$$

$res\_pattern\_jtype$ ::=

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

$ret\_pattern\_jtype$ ::=

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

$tpval\_jtype$ ::=

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

$tpexpr\_jtype$ ::=

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

$action\_jtype$ ::=

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

$memop\_jtype$ ::=

    | $\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem\_op \Rightarrow ret$

$tval\_jtype$ ::=

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

seq_expr_jtype ::=

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

is_expr_jtype ::=

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

$texpr\_jtype$ ::=

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

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

$subs\_jtype$ ::=

$$| \quad pattern = pval \rightsquigarrow \sigma$$
$$| \quad ident\_or\_pattern = pval \rightsquigarrow \sigma$$
$$| \quad res\_pattern = res\_term \rightsquigarrow \sigma$$
$$| \quad \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma$$
$$| \quad \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret$$

$pure\_opsem\_jtype$ ::=

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

$opsem\_jtype$ ::=

$$| \quad \langle h; seq\_expr \rangle \longrightarrow \langle h'; texpr{:}ret \rangle$$
$$| \quad \langle h; seq\_texpr \rangle \longrightarrow \langle h'; texpr \rangle$$
$$| \quad \langle h; mem\_op \rangle \longrightarrow \langle h'; tval \rangle$$
$$| \quad \langle h; mem\_action \rangle \longrightarrow \langle h'; tval \rangle$$
$$| \quad \langle h; is\_expr \rangle \longrightarrow \langle h'; is\_expr' \rangle$$
$$| \quad \langle h; is\_texpr \rangle \longrightarrow \langle h'; texpr \rangle$$
$$| \quad \langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle$$

$lemma\_jtype$ ::=

$$| \quad \overline{x_i}^{\,i} :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret$$
$$| \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'$$
$$| \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$$

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

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash mem\_int \Rightarrow \mathtt{obj}\ \mathtt{integer}} \quad \text{TY\_PVAL\_OBJ\_INT}$$

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\overline{\mathcal{C}; \mathcal{L}; \Phi \vdash value_i \Rightarrow \beta}^{\,i}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \beta[\overline{value_i}^{\,i}] \Rightarrow \texttt{list}\,\beta} \quad \text{Ty\_Pval\_List}$$

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

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

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \beta \quad \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \texttt{list}\,\beta}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{Cons}(pval_1, pval_2) \Rightarrow \texttt{list}\,\beta} \quad \text{Ty\_Pval\_Ctor\_Cons}$$

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

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

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

$$\dfrac{\begin{array}{c} ident\!:\!\texttt{struct}\,tag\;\&\;\overline{member_i\!:\!\tau_i}^{\,i} \in \texttt{Globals} \\[4pt] \overline{\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow \beta_{\tau_i}}^{\,i} \end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash (\,\texttt{struct}\,tag)\{\,\overline{.\,member_i = pval_i}^{\,i}\,\} \Rightarrow \texttt{struct}\,tag}\quad \textsc{Ty\_Pval\_Struct}$$

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

$$\dfrac{}{\Phi \vdash \texttt{emp} \equiv \texttt{emp}}\quad \textsc{Ty\_Res\_Eq\_Emp}$$

$$\dfrac{\texttt{smt}\,(\Phi \Rightarrow (term_1 = term_1') \wedge (term_2 = term_2'))}{\Phi \vdash term_1 \overset{init}{\mapsto}_\tau term_2 \equiv term_1' \overset{init}{\mapsto}_\tau term_2'}\quad \textsc{Ty\_Res\_Eq\_PointsTo}$$

$$\dfrac{\begin{array}{c}\Phi \vdash res_1 \equiv res_1' \\[4pt] \Phi \vdash res_2 \equiv res_2'\end{array}}{\Phi \vdash res_1 * res_2 \equiv res_1' * res_2'}\quad \textsc{Ty\_Res\_Eq\_SepConj}$$

$$\dfrac{\Phi \vdash res \equiv res'}{\Phi \vdash \exists ident\!:\!\beta.\;res \equiv \exists ident\!:\!\beta.\;res'}\quad \textsc{Ty\_Res\_Eq\_Exists}$$

$$\dfrac{\begin{array}{c}\texttt{smt}\,(\Phi, term \Rightarrow term') \\[4pt] \texttt{smt}\,(\Phi, term' \Rightarrow term) \\[4pt] \Phi \vdash res \equiv res'\end{array}}{\Phi \vdash term \wedge res \equiv term' \wedge res'}\quad \textsc{Ty\_Res\_Eq\_Term}$$

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

$$\dfrac{}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \texttt{emp} \Leftarrow \texttt{emp}}\quad \textsc{Ty\_Res\_Emp}$$

$$\frac{\Phi \vdash points\_to \equiv points\_to'}{\mathcal{C}; \mathcal{L}; \Phi; \cdot, points\_to \vdash points\_to' \Leftarrow points\_to'} \quad \text{Ty\_Res\_PointsTo}$$

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

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

$$\frac{\begin{array}{c} \mathtt{smt}\,(\Phi \Rightarrow term) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Leftarrow res \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Leftarrow term \wedge res} \quad \text{Ty\_Res\_Conj}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term_2 \Leftarrow pval/y, \cdot(res) \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{pack}\,(pval, res\_term_2) \Leftarrow \exists\,y{:}\beta.\ res} \quad \text{Ty\_Res\_Pack}$$

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

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

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

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

23

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

$$\frac{\mathtt{smt}\,(\Phi \Rightarrow term) \qquad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: term \supset arg \gg \sigma; ret} \quad \text{Ty\_Spine\_Phi}$$

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

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

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

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

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \mathtt{integer} \qquad \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \mathtt{integer}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1\ binop_{arith}\ pval_2 \Rightarrow y{:}\mathtt{integer}.\ y = (pval_1\ binop_{arith}\ pval_2)} \quad \text{Ty\_PE\_Arith\_Binop}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \texttt{integer} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \texttt{integer} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \ binop_{rel} \ pval_2 \Rightarrow y{:}\texttt{bool}.\ y = (pval_1 \ binop_{rel} \ pval_2)} \quad \text{TY\_PE\_REL\_BINOP}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \texttt{bool} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \texttt{bool} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \ binop_{bool} \ pval_2 \Rightarrow y{:}\texttt{bool}.\ y = (pval_1 \ binop_{bool} \ pval_2)} \quad \text{TY\_PE\_BOOL\_BINOP}$$

$$\frac{\begin{array}{c} name{:}pure\_arg \equiv \overline{x_i}^{\,i} \mapsto tpexpr \ \in \ \texttt{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \overline{x_i = pval_i}^{\,i} :: pure\_arg \gg \sigma; \Sigma \, y{:}\beta.\ term \wedge \texttt{I} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash name(\overline{pval_i}^{\,i}) \Rightarrow y{:}\beta.\ \sigma(term)} \quad \text{TY\_PE\_CALL}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \texttt{bool} \\ \texttt{smt}\,(\Phi \Rightarrow pval) \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{assert\_undef}\,(pval,\ UB\_name) \Rightarrow y{:}\texttt{unit}.\ y = \texttt{unit}} \quad \text{TY\_PE\_ASSERT\_UNDEF}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \texttt{bool}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{bool\_to\_integer}\,(pval) \Rightarrow y{:}\texttt{integer}.\ y = \texttt{if}\ pval\ \texttt{then}\ 1\ \texttt{else}\ 0} \quad \text{TY\_PE\_BOOL\_TO\_INTEGER}$$

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

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

$$\frac{}{\_{:}\beta{:}\beta \rightsquigarrow \cdot \ \texttt{with}\ \_} \quad \text{TY\_PAT\_COMP\_NO\_SYM\_ANNOT}$$

$$\overline{x{:}\beta{:}\beta \rightsquigarrow \cdot, x{:}\beta \,\texttt{with}\, x} \quad \textsc{Ty\_Pat\_Comp\_Sym\_Annot}$$

$$\overline{\texttt{Nil}\,\beta(\,){:}\texttt{list}\,\beta \rightsquigarrow \cdot\,\texttt{with}\,\texttt{nil}} \quad \textsc{Ty\_Pat\_Comp\_Nil}$$

$$\frac{\begin{array}{c} pattern_1{:}\beta \rightsquigarrow \mathcal{C}_1 \,\texttt{with}\, term_1 \\ pattern_2{:}\texttt{list}\,\beta \rightsquigarrow \mathcal{C}_2 \,\texttt{with}\, term_2 \end{array}}{\texttt{Cons}(pattern_1, pattern_2){:}\texttt{list}\,\beta \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2 \,\texttt{with}\, term_1 :: term_2} \quad \textsc{Ty\_Pat\_Comp\_Cons}$$

$$\frac{\overline{pattern_i{:}\beta_i \rightsquigarrow \mathcal{C}_i \,\texttt{with}\, term_i}^{\,i}}{\texttt{Tuple}(\overline{pattern_i}^{\,i}){:}\overline{\beta_i}^{\,i} \rightsquigarrow \overline{\mathcal{C}_i}^{\,i} \,\texttt{with}\, (\overline{term_i}^{\,i})} \quad \textsc{Ty\_Pat\_Comp\_Tuple}$$

$$\frac{\overline{pattern_i{:}\beta \rightsquigarrow \mathcal{C}_i \,\texttt{with}\, term_i}^{\,i}}{\texttt{Array}(\overline{pattern_i}^{\,i}){:}\texttt{array}\,\beta \rightsquigarrow \overline{\mathcal{C}_i}^{\,i} \,\texttt{with}\, [|\ \overline{term_i}^{\,i}\ |]} \quad \textsc{Ty\_Pat\_Comp\_Array}$$

$$\frac{pattern{:}\beta \rightsquigarrow \mathcal{C} \,\texttt{with}\, term}{\texttt{Specified}(pattern){:}\beta \rightsquigarrow \mathcal{C} \,\texttt{with}\, term} \quad \textsc{Ty\_Pat\_Comp\_Specified}$$

$$\boxed{ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C} \,\texttt{with}\, term}$$

$$\overline{x{:}\beta \rightsquigarrow \cdot, x{:}\beta \,\texttt{with}\, x} \quad \textsc{Ty\_Pat\_Sym\_Or\_Pattern\_Sym}$$

$$\frac{pattern{:}\beta \rightsquigarrow \mathcal{C} \,\texttt{with}\, term}{pattern{:}\beta \rightsquigarrow \mathcal{C} \,\texttt{with}\, term} \quad \textsc{Ty\_Pat\_Sym\_Or\_Pattern\_Pattern}$$

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

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

$$\frac{}{points\_to{:}points\_to \rightsquigarrow \cdot;\cdot;\cdot, points\_to} \quad \text{Ty\_Pat\_Res\_PointsTo}$$

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

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

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

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

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

$$\frac{}{{:}\mathrm{I} \rightsquigarrow \cdot;\cdot;\cdot;\cdot} \quad \text{Ty\_Pat\_Ret\_Empty}$$

$$\frac{\begin{array}{c} ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1\ \texttt{with}\ term_1 \\ \overline{ret\_pattern_i}^{\,i}{:}term_1/y,\cdot(ret) \rightsquigarrow \mathcal{C}_2;\mathcal{L}_2;\Phi_2;\mathcal{R}_2 \end{array}}{\texttt{comp}\,ident\_or\_pattern, \overline{ret\_pattern_i}^{\,i}{:}\Sigma\, y{:}\beta.\ ret \rightsquigarrow \mathcal{C}_1,\mathcal{C}_2;\mathcal{L}_2;\Phi_1,\Phi_2;\mathcal{R}_2} \quad \text{Ty\_Pat\_Ret\_Comp}$$

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

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

$$\frac{\overline{ret\_pattern_i}^{\,i}{:}ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\overline{ret\_pattern_i}^{\,i}{:}term \wedge ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi, term; \mathcal{R}} \quad \text{Ty\_Pat\_Ret\_Phi}$$

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

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

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

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

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

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y_1{:}\beta_1.\ term_1 \\ ident\_or\_pattern{:}\beta_1 \rightsquigarrow \mathcal{C}_1\ \mathtt{with}\ term \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term/y_1, \cdot(term_1), \Phi_1 \vdash tpexpr \Leftarrow y_2{:}\beta_2.\ term_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathtt{let}\ ident\_or\_pattern = pexpr\ \mathtt{in}\ tpexpr \Leftarrow y_2{:}\beta_2.\ term_2} \quad \text{Ty\_TPE\_Let}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr_1 \Leftarrow y_1{:}\beta_1.\ term_1 \\ ident\_or\_pattern{:}\beta_1 \rightsquigarrow \mathcal{C}_1 \ \texttt{with}\ term \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term/y_1, \cdot(term_1), \Phi_1 \vdash tpexpr \Leftarrow y_2{:}\beta_2.\ term_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{let}\ ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr_1\ \texttt{in}\ tpexpr_2 \Leftarrow y_2{:}\beta_2.\ term_2} \quad \text{TY\_TPE\_LETT}$$

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta_1 \\ \overline{pattern_i{:}\beta_1 \rightsquigarrow \mathcal{C}_i \ \texttt{with}\ term_i}^{\ i} \\ \overline{\mathcal{C}, \mathcal{C}_i; \mathcal{L}; \Phi, term_i = pval, \Phi_i \vdash tpexpr_i \Leftarrow y_2{:}\beta_2.\ term_2}^{\ i} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{case}\ pval\ \texttt{of}\ \overline{\mid pattern_i \Rightarrow tpexpr_i}^{\ i}\ \texttt{end} \Leftarrow y_2{:}\beta_2.\ term_2} \quad \text{TY\_TPE\_CASE}$$

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \texttt{integer}}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash \texttt{create}\,(pval, \tau) \Rightarrow \Sigma\, y_p{:}\texttt{loc}.\ \texttt{representable}\,(\tau*, y_p) \wedge \texttt{alignedI}\,(pval, y_p) \wedge \exists\, y{:}\beta_\tau.\ y_p \overset{\times}{\mapsto}_\tau y \otimes \texttt{I}} \quad \text{TY\_ACTION\_CREATE}$$

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

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \texttt{loc} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \beta_\tau \\ \texttt{smt}\,(\Phi \Rightarrow \texttt{representable}\,(\tau, pval_1)) \\ \texttt{smt}\,(\Phi \Rightarrow pval_2 = pval_0) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_2 \mapsto_\tau \_ \Leftarrow pval_2 \mapsto_\tau \_ \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{store}\,(\_, \tau, pval_0, pval_1, \_, pval_2 \mapsto_\tau \_) \Rightarrow \Sigma\, \_{:}\texttt{unit}.\ pval_2 \overset{\checkmark}{\mapsto}_\tau pval_1 \otimes \texttt{I}} \quad \text{TY\_ACTION\_STORE}$$

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_0 \Rightarrow \texttt{loc} \\ \texttt{smt}\,(\Phi \Rightarrow pval_0 = pval_1) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash pval_1 \mapsto_{\tau\,\_} \Leftarrow pval_1 \mapsto_{\tau\,\_} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{kill}\,(\texttt{static}\,\tau, pval_0, pval_1 \mapsto_{\tau\,\_}) \Rightarrow \Sigma\,\_:\texttt{unit}.\,\texttt{I}} \quad \text{Ty\_Action\_Kill\_Static}$$

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

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pval_1 \Rightarrow \texttt{loc} \\ \mathcal{C}; \mathcal{L}; \Phi \vdash pval_2 \Rightarrow \texttt{loc} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \cdot \vdash pval_1\,binop_{rel}\,pval_2 \Rightarrow \Sigma\,y:\texttt{bool}.\,y = (pval_1\,binop_{rel}\,pval_2) \wedge \texttt{I}} \quad \text{Ty\_Memop\_Rel\_Binop}$$

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

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

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

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

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

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \mathtt{done} \ \Leftarrow \mathtt{I}} \ \ \text{TY\_TVAL\_I}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{done} \ \overline{spine\_elem_i}^{\ i} \Leftarrow pval/y, \cdot(ret) \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{done} \ pval, \ \overline{spine\_elem_i}^{\ i} \Leftarrow \Sigma \ y{:}\beta. \ ret} \ \ \text{TY\_TVAL\_COMP}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{done} \ \overline{spine\_elem_i}^{\ i} \Leftarrow pval/y, \cdot(ret) \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{done} \ pval, \ \overline{spine\_elem_i}^{\ i} \Leftarrow \exists \ y{:}\beta. \ ret} \ \ \text{TY\_TVAL\_LOG}$$

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

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow res \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_2 \vdash \mathtt{done} \ \overline{spine\_elem_i}^{\ i} \Leftarrow ret \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1,\mathcal{R}_2 \vdash \mathtt{done} \ res\_term, \ \overline{spine\_elem}^{\ i} \Leftarrow res \otimes ret} \ \ \text{TY\_TVAL\_RES}$$

$$\frac{\mathtt{smt} \ (\Phi \Rightarrow \mathtt{false})}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \mathtt{undef} \ UB\_name \Leftarrow ret} \ \ \text{TY\_TVAL\_UB}$$

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

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

$$\dfrac{\begin{array}{c} name{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \mathtt{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{pcall}\,(name, \overline{spine\_elem_i}^{\,i}) \Rightarrow \sigma(ret)} \quad \text{Ty\_Seq\_E\_Proc}$$

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

$$\dfrac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem\_op \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{memop}\,(mem\_op) \Rightarrow ret} \quad \text{Ty\_Is\_E\_Memop}$$

$$\dfrac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem\_action \Rightarrow ret} \quad \text{Ty\_Is\_E\_Action}$$

$$\dfrac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{neg}\, mem\_action \Rightarrow ret} \quad \text{Ty\_Is\_E\_Neg\_Action}$$

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

$$\dfrac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash tval \Leftarrow ret} \quad \text{Ty\_Seq\_TE\_TVal}$$

$$\dfrac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow y{:}\beta.\ term \\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1 \,\mathtt{with}\, term_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term_1/y, \cdot(term), \Phi_1; \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{let}\, ident\_or\_pattern = pexpr \,\mathtt{in}\, texpr \Leftarrow ret} \quad \text{Ty\_Seq\_TE\_LetP}$$

$$\dfrac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr \Leftarrow y{:}\beta.\ term \\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1 \,\mathtt{with}\, term_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}; \Phi, term_1/y, \cdot(term), \Phi_1; \mathcal{R} \vdash texpr \Leftarrow ret \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{let}\, ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr \,\mathtt{in}\, texpr \Leftarrow ret} \quad \text{Ty\_Seq\_TE\_LetPT}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash seq\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i}\!:ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash texpr \Leftarrow ret_2 \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}',\mathcal{R} \vdash \mathtt{let}\,\overline{ret\_pattern_i}^{\,i} = seq\_expr\,\mathtt{in}\,texpr \Leftarrow ret_2} \quad \text{TY\_SEQ\_TE\_LET}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash texpr_1 \Leftarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i}\!:ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathcal{C}_1;\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash texpr_2 \Leftarrow ret_2 \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}',\mathcal{R} \vdash \mathtt{let}\,\overline{ret\_pattern_i}^{\,i}\!:ret_1 = texpr_1\,\mathtt{in}\,texpr_2 \Leftarrow ret_2} \quad \text{TY\_SEQ\_TE\_LETT}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta_1 \\ \overline{pattern_i\!:\!\beta_1 \rightsquigarrow \mathcal{C}_i\,\mathtt{with}\,term_i}^{\,i} \\ \overline{\mathcal{C},\mathcal{C}_i;\mathcal{L};\Phi,term_i = pval,\Phi_i;\mathcal{R} \vdash texpr_i \Leftarrow ret}^{\,i} \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{case}\,pval\,\mathtt{of}\,\overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\,\mathtt{end} \Leftarrow ret} \quad \text{TY\_SEQ\_TE\_CASE}$$

$$\frac{\begin{array}{c} \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \mathtt{bool} \\ \mathcal{C};\mathcal{L};\Phi,pval = \mathtt{true};\mathcal{R} \vdash texpr_1 \Leftarrow ret \\ \mathcal{C};\mathcal{L};\Phi,pval = \mathtt{false};\mathcal{R} \vdash texpr_2 \Leftarrow ret \end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{if}\,pval\,\mathtt{then}\,texpr_1\,\mathtt{else}\,texpr_2 \Leftarrow ret} \quad \text{TY\_SEQ\_TE\_IF}$$

$$\frac{\begin{array}{c} ident\!:\!arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \mathtt{Globals} \\ \mathcal{C};\mathcal{L};\Phi;\cdot \vdash \overline{x_i = pval_i}^{\,i} :: arg \gg \sigma;\mathtt{false} \wedge \mathtt{I} \end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \mathtt{run}\,ident\,\overline{pval_i}^{\,i} \Leftarrow \mathtt{false} \wedge \mathtt{I}} \quad \text{TY\_SEQ\_TE\_RUN}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_texpr \Leftarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{bound}\,[int](is\_texpr) \Leftarrow ret} \quad \text{TY\_SEQ\_TE\_BOUND}$$

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

33

$$\frac{\begin{array}{c} \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1; \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ \mathcal{C}, \mathcal{C}_1; \mathcal{L}, \mathcal{L}_1; \Phi, \Phi_1; \mathcal{R}, \mathcal{R}_1 \vdash texpr \Leftarrow ret_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{let strong} \, \overline{ret\_pattern_i}^{\,i} = is\_expr \, \texttt{in} \, texpr \Leftarrow ret_2} \quad \text{TY\_IS\_TE\_LETS}$$

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_texpr \Leftarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_texpr \Leftarrow ret} \quad \text{TY\_TE\_IS}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq\_texpr \Leftarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq\_texpr \Leftarrow ret} \quad \text{TY\_TE\_SEQ}$$

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

$$\frac{}{\_:\_ = pval \rightsquigarrow \cdot} \quad \text{SUBS\_DECONS\_VALUE\_NO\_SYM\_ANNOT}$$

$$\frac{}{x:\_ = pval \rightsquigarrow pval/x, \cdot} \quad \text{SUBS\_DECONS\_VALUE\_SYM\_ANNOT}$$

$$\frac{\begin{array}{c} pattern_1 = pval_1 \rightsquigarrow \sigma_1 \\ pattern_2 = pval_2 \rightsquigarrow \sigma_2 \end{array}}{\texttt{Cons}(pattern_1, pattern_2) = \texttt{Cons}(pval_1, pval_2) \rightsquigarrow \sigma_1, \sigma_2} \quad \text{SUBS\_DECONS\_VALUE\_CONS}$$

$$\frac{\overline{pattern_i = pval_i \rightsquigarrow \sigma_i}^{\,i}}{\texttt{Tuple}(\overline{pattern_i}^{\,i}) = \texttt{Tuple}(\overline{pval_i}^{\,i}) \rightsquigarrow \overline{\sigma_i}^{\,i}} \quad \text{SUBS\_DECONS\_VALUE\_TUPLE}$$

34

$$\frac{\overline{pattern_i = pval_i \rightsquigarrow \sigma_i}^{\,i}}{\mathtt{Array}(\overline{pattern_i}^{\,i}) = \mathtt{Array}(\overline{pval_i}^{\,i}) \rightsquigarrow \overline{\sigma_i}^{\,i}} \quad \text{\textsc{Subs\_Decons\_Value\_Array}}$$

$$\frac{pattern = pval \rightsquigarrow \sigma}{\mathtt{Specified}(pattern) = pval \rightsquigarrow \sigma} \quad \text{\textsc{Subs\_Decons\_Value\_Specified}}$$

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

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

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

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

$$\frac{}{\mathtt{emp} = \mathtt{emp} \rightsquigarrow \cdot} \quad \text{\textsc{Subs\_Decons\_Res\_Emp}}$$

$$\frac{}{pt = pt \rightsquigarrow \cdot} \quad \text{\textsc{Subs\_Decons\_Res\_Points\_to}}$$

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

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

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

$$\boxed{\overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma}$$

$$\frac{}{\rightsquigarrow \cdot} \quad \text{Subs\_Decons\_Ret\_Empty}$$

$$\frac{\begin{array}{c} ident\_or\_pattern = pval \rightsquigarrow \sigma \\ \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \psi \end{array}}{\texttt{comp}\,ident\_or\_pattern = pval, \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma, \psi} \quad \text{Subs\_Decons\_Ret\_Comp}$$

$$\frac{\overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \psi}{\texttt{log}\,ident = pval, \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow pval/ident, \psi} \quad \text{Subs\_Decons\_Ret\_Log}$$

$$\frac{\begin{array}{c} res\_pattern = res\_term \rightsquigarrow \sigma \\ \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \psi \end{array}}{\texttt{res}\,res\_pattern = res\_term, \overline{ret\_pattern_i = spine\_elem_i}^{\,i} \rightsquigarrow \sigma, \psi} \quad \text{Subs\_Decons\_Ret\_Res}$$

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

$$\frac{}{:: ret \gg \cdot; ret} \quad \text{Subs\_Decons\_Arg\_Empty}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine\_elem_i}^{\,i} :: \Pi\,x{:}\beta.\ arg \gg pval/x, \sigma; ret} \quad \text{Subs\_Decons\_Arg\_Comp}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine\_elem_i}^{\,i} :: \forall\,x{:}\beta.\ arg \gg pval/x, \sigma; ret} \quad \text{Subs\_Decons\_Arg\_Log}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = res\_term, \; \overline{x_i = spine\_elem_i}^{\,i} :: res \multimap arg \gg res\_term/x, \sigma; ret} \quad \textsc{Subs\_Decons\_Arg\_Res}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{\overline{x_i = spine\_elem_i}^{\,i} :: term \supset arg \gg \sigma; ret} \quad \textsc{Subs\_Decons\_Arg\_Phi}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c} name{:}pure\_arg \equiv \overline{x_i}^{\,i} \mapsto tpexpr \, \in \, \texttt{Globals} \\ \overline{x_i = pval_i}^{\,i} :: pure\_arg \gg \sigma; \Sigma \, y{:}\beta.\, term \wedge \texttt{I} \end{array}}{\langle name(\overline{pval_i}^{\,i}) \rangle \longrightarrow \langle \sigma(tpexpr){:}(y{:}\beta.\, \sigma(term)) \rangle} \quad \text{Op\_PE\_TPE\_Call}$$

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

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

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

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

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

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

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

$$\dfrac{}{\langle \mathtt{if\ True\ then}\ tpexpr_1\ \mathtt{else}\ tpexpr_2 \rangle \longrightarrow \langle tpexpr_1 \rangle} \quad \textsc{Op\_TPE\_TPE\_If\_True}$$

$$\dfrac{}{\langle \mathtt{if\ False\ then}\ tpexpr_1\ \mathtt{else}\ tpexpr_2 \rangle \longrightarrow \langle tpexpr_2 \rangle} \quad \textsc{Op\_TPE\_TPE\_If\_False}$$

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

$$\dfrac{\begin{array}{c} mem\_ptr{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \mathtt{Globals} \\ \overline{x_i = spine\_elem_i}^{\,i} {::}\ arg \gg \sigma; ret \end{array}}{\langle h; \mathtt{ccall}\,(\tau, mem\_ptr, \overline{spine\_elem_i}^{\,i}) \rangle \longrightarrow \langle h; \sigma(texpr){:}\sigma(ret) \rangle} \quad \textsc{Op\_SE\_TE\_CCall}$$

$$\dfrac{\begin{array}{c} name{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \mathtt{Globals} \\ \overline{x_i = spine\_elem_i}^{\,i} {::}\ arg \gg \sigma; ret \end{array}}{\langle h; \mathtt{pcall}\,(name, \overline{spine\_elem_i}^{\,i}) \rangle \longrightarrow \langle h; \sigma(texpr){:}\sigma(ret) \rangle} \quad \textsc{Op\_SE\_TE\_PCall}$$

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

$$\dfrac{\begin{array}{c} ident{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \mathtt{Globals} \\ \overline{x_i = pval_i}^{\,i} {::}\ arg \gg \sigma; \mathtt{false} \wedge \mathtt{I} \end{array}}{\langle h; \mathtt{run}\ ident\ \overline{pval_i}^{\,i} \rangle \longrightarrow \langle h; \sigma(texpr) \rangle} \quad \textsc{Op\_STE\_TE\_Run}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{bool\_value \equiv mem\_int_1 \; binop_{rel} \; mem\_int_2}{\langle h; mem\_int_1 \; binop_{rel} \; mem\_int_2 \rangle \longrightarrow \langle h; \texttt{done } bool\_value \rangle} \quad \text{Op\_Memop\_TVal\_Rel\_Binop}$$

$$\frac{mem\_int \equiv \texttt{cast\_ptr\_to\_int } mem\_ptr}{\langle h; \texttt{intFromPtr} \, (\tau_1, \tau_2, mem\_ptr) \rangle \longrightarrow \langle h; \texttt{done } mem\_int \rangle} \quad \text{Op\_Memop\_TVal\_IntFromPtr}$$

$$\frac{mem\_ptr \equiv \texttt{cast\_ptr\_to\_int } mem\_int}{\langle h; \texttt{ptrFromInt} \, (\tau_1, \tau_2, mem\_int) \rangle \longrightarrow \langle h; \texttt{done } mem\_ptr \rangle} \quad \text{Op\_Memop\_TVal\_PtrFromInt}$$

$$\frac{bool\_value \equiv \texttt{aligned} \, (\tau, mem\_ptr)}{\langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_\}; \texttt{ptrValidForDeref} \, (\tau, mem\_ptr, mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_) \rangle \longrightarrow \langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_\}; \texttt{done } bool\_value, mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_ \rangle} \quad \text{Op\_Memop\_TVal\_PtrValid}$$

$$\frac{bool\_value \equiv \texttt{aligned} \, (\tau, mem\_ptr)}{\langle h; \texttt{ptrWellAligned} \, (\tau, mem\_ptr) \rangle \longrightarrow \langle h; \texttt{done } bool\_value \rangle} \quad \text{Op\_Memop\_TVal\_PtrWellAligned}$$

$$\frac{mem\_ptr' \equiv mem\_ptr +_{\text{ptr}} (mem\_int \times \text{size\_of}(\tau))}{\langle h; \texttt{ptrArrayShift} \, (mem\_ptr, \tau, mem\_int) \rangle \longrightarrow \langle h; \texttt{done } mem\_ptr' \rangle} \quad \text{Op\_Memop\_TVal\_PtrArrayShift}$$

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

$$\frac{\begin{array}{c} \texttt{fresh}\,(mem\_ptr) \\ \texttt{representable}\,(\tau*, mem\_ptr) \\ \texttt{alignedI}\,(mem\_int, mem\_ptr) \\ pval{:}\beta_\tau \end{array}}{\langle h; \texttt{create}\,(mem\_int, \tau) \rangle \longrightarrow \langle h + \{mem\_ptr \overset{\times}{\mapsto}_\tau pval\}; \texttt{done}\, mem\_ptr, pval, mem\_ptr \overset{\times}{\mapsto}_\tau pval \rangle} \quad \textsc{Op\_Action\_Tval\_Create}$$

$$\frac{}{\langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval\}; \texttt{load}\,(\tau, mem\_ptr, \_, mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval) \rangle \longrightarrow \langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval\}; \texttt{done}\, pval, mem\_ptr \overset{\checkmark}{\mapsto}_\tau pval \rangle} \quad \textsc{Op\_Action\_Tval\_Load}$$

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

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

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

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

$$\frac{\langle h; mem\_action \rangle \longrightarrow \langle h'; tval \rangle}{\langle h; mem\_action \rangle \longrightarrow \langle h'; tval \rangle} \quad \textsc{Op\_IsE\_IsE\_Action}$$

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

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

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

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

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

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

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

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

$$\frac{}{::ret \rightsquigarrow \cdot; \cdot; \cdot; \cdot \mid ret}\quad \text{Arg\_Env\_Ret}$$

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

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

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

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

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

$$\frac{}{\cdot;\cdot;\cdot;\cdot \sqsubseteq \cdot;\cdot;\cdot;\cdot} \quad \text{WEAK\_EMPTY}$$

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

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

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

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

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

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

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

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash (\cdot){:}(\cdot;\cdot;\cdot;\cdot)} \quad \text{TY\_SUBS\_EMPTY}$$

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}')}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi', term;\mathcal{R}')} \quad \text{TY\_SUBS\_CONS\_PHI}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow res\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}, \mathcal{R}_1 \vdash (res\_term/x, \sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}', x{:}res)} \quad \text{TY\_SUBS\_CONS\_RES\_NAMED}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \mathcal{C};\mathcal{L};\Phi;\overline{\cdot, res_i}^{\,i} \vdash res\_term \Leftarrow res\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}, \overline{\cdot, res_i}^{\,i} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}', \overline{\cdot, res_i}^{\,i})} \quad \text{TY\_SUBS\_CONS\_RES\_ANON}$$

```
Definition rules:        199 good     0 bad
Definition rule clauses: 444 good     0 bad
```