ident, x, y, y_p, y_f, _, *abbrev*, r, α	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

	| `!=`	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
	|	`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
	_: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	::=		
	| $\neg\, term$		
	| $term_1 = term_2$		
	| $term_1 \rightarrow term_2$		
	| $\bigwedge(\overline{term_i}^{\,i})$		
	| $\bigvee(\overline{term_i}^{\,i})$		
	| $term_1\ binop_{bool}\ term_2$	M	
	| `if` $term_1$ `then` $term_2$ `else` $term_3$		
arith_op	::=		
	| $term_1 + term_2$		
	| $term_1 - term_2$		
	| $term_1 \times term_2$		
	| $term_1/term_2$		
	| $term_1$ `rem_t` $term_2$		
	| $term_1$ `rem_f` $term_2$		
	| $term_1$ `^` $term_2$		
	| $term_1\ binop_{arith}\ term_2$	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	::=		
	| `nil`		

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

$term,\ \_$	::=			
	|	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
	| $\texttt{case}\, pval\, \texttt{of}\, \overline{\mid tpexpr\_case\_branch_i}^{\,i}\, \texttt{end}$		pattern matching
	| $\texttt{let}\, ident\_or\_pattern = pexpr\, \texttt{in}\, tpexpr$	bind binders($ident\_or\_pattern$) in $tpexpr$	pure let
	| $\texttt{let}\, ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr_1\, \texttt{in}\, tpexpr_2$	bind binders($ident\_or\_pattern$) in $tpexpr_2$ bind y_1 in $term_1$	annoted pure let
	| $\texttt{if}\, pval\, \texttt{then}\, tpexpr_1\, \texttt{else}\, tpexpr_2$		pure if
	| $\sigma(tpexpr)$	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`		
	| $\texttt{static}\, \tau$		
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
	| $\texttt{pack}\, (pval, res\_term)$		packing for existentials
	| $\texttt{fold}\, (res\_term)$		fold into recursive res. pred

	$\mid$	$\sigma(res\_term)$	M substitution for resource terms
mem_action	::=		memory actions
	$\mid$	$\texttt{create}\,(pval, \tau)$	
	$\mid$	$\texttt{create\_readonly}\,(pval_1, \tau, pval_2)$	
	$\mid$	$\texttt{alloc}\,(pval_1, pval_2)$	
	$\mid$	$\texttt{kill}\,(m\_kill\_kind, pval, pt)$	
	$\mid$	$\texttt{store}\,(bool, \tau, pval_1, pval_2, mem\_order, pt)$	true means store is locking
	$\mid$	$\texttt{load}\,(\tau, pval, mem\_order, pt)$	
	$\mid$	$\texttt{rmw}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	$\mid$	$\texttt{fence}\,(mem\_order)$	
	$\mid$	$\texttt{cmp\_exch\_strong}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	$\mid$	$\texttt{cmp\_exch\_weak}\,(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$	
	$\mid$	$\texttt{linux\_fence}\,(linux\_mem\_order)$	
	$\mid$	$\texttt{linux\_load}\,(\tau, pval, linux\_mem\_order)$	
	$\mid$	$\texttt{linux\_store}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	
	$\mid$	$\texttt{linux\_rmw}\,(\tau, pval_1, pval_2, linux\_mem\_order)$	
$polarity$	::=		polarities for memory actions
	$\mid$		(pos) sequenced by `let weak` and `let strong`
	$\mid$	`neg`	only sequenced by `let strong`
pol_mem_action	::=		memory actions with polarity
	$\mid$	$polarity\; mem\_action$	
mem_op	::=		operations involving the memory state
	$\mid$	$pval_1\ binop_{rel}\ pval_2$	pointer relational binary operations
	$\mid$	$pval_1 -_\tau pval_2$	pointer subtraction
	$\mid$	$\texttt{intFromPtr}\,(\tau_1, \tau_2, pval)$	cast of pointer value to integer value
	$\mid$	$\texttt{ptrFromInt}\,(\tau_1, \tau_2, pval)$	cast of integer value to pointer value

	$\texttt{ptrValidForDeref}\,(\tau, pval, pt)$		dereferencing validity predicate
	$\texttt{ptrWellAligned}\,(\tau, pval)$		
	$\texttt{ptrArrayShift}\,(pval_1, \tau, pval_2)$		
	$\texttt{memcpy}\,(pval_1, pval_2, pval_3)$		
	$\texttt{memcmp}\,(pval_1, pval_2, pval_3)$		
	$\texttt{realloc}\,(pval_1, pval_2, pval_3)$		
	$\texttt{va\_start}\,(pval_1, pval_2)$		
	$\texttt{va\_copy}\,(pval)$		
	$\texttt{va\_arg}\,(pval, \tau)$		
	$\texttt{va\_end}\,(pval)$		
$spine\_elem$ ::=			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
	$\texttt{done}\ spine$		end of top-level expression
	$\texttt{undef}\ UB\_name$		undefined behaviour
$res\_pattern$ ::=			resource terms
	$\texttt{emp}$	$\text{binders} = \{\}$	empty heap
	$ident$	$\text{binders} = ident$	variable
	$\texttt{fold}\,(res\_pattern)$	$\text{binders} = \{\}$	unfold (recursive) predicate
	$\langle res\_pattern_1, res\_pattern_2 \rangle$	$\text{binders} = \text{binders}(res\_pattern_1) \cup \text{binders}(res\_pattern_2)$	seperating-conjunction pair
	$\texttt{pack}\,(ident, res\_pattern)$	$\text{binders} = ident \cup \text{binders}(res\_pattern)$	packing for existentials

ret_pattern	::=			return pattern
	|	`comp` *ident_or_pattern*	binders = binders(*ident_or_pattern*)	computational variable
	|	`log` *ident*	binders = *ident*	logical variable
	|	`res` *res_pattern*	binders = binders(*res_pattern*)	resource variable
init,	::=			initialisation status
	|	✓		initialised
	|	×		uninitalised
points_to, *pt*	::=			points-to separation logic predicate
	|	$term_1 \overset{init}{\mapsto}_\tau term_2$		
res	::=			resources
	|	`emp`		empty heap
	|	*points_to*		points-top heap pred.
	|	$res_1 * res_2$		seperating conjunction
	|	$\exists\, ident{:}\beta.\ res$		existential
	|	$term \wedge res$		logical conjuction
	|	`if` $term$ `then` res_1 `else` res_2		ordered disjuction
	|	$\alpha(\overline{pval_i}^{\,i})$		predicate
	|	$\sigma(res)$	M	simul-sub σ in *res*
ret, _	::=			return types
	|	$\Sigma\, ident{:}\beta.\ ret$		return a computational value
	|	$\exists\, ident{:}\beta.\ ret$		return a logical value
	|	$res \otimes ret$		return a resource value
	|	$term \wedge ret$		return a predicate (post-condition)
	|	`I`		end return list
	|	$\sigma(ret)$	M	simul-sub σ in *ret*

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

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

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

	|	$pval{:}\beta$
formula	::=	
	|	*judgement*
	|	*typing*
	|	*opsem*
	|	$term \equiv term'$
	|	$name{:}pure\_arg \equiv \overline{x_i}^{\,i} \mapsto tpexpr \in \texttt{Globals}$
	|	$name{:}arg \equiv \overline{x_i}^{\,i} \mapsto texpr \in \texttt{Globals}$
heap, h	::=	heaps
	|	.
	|	$h + \{points\_to\}$
lemma_jtype	::=	
	|	$\overline{x_i}^{\,i} :: arg \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \mid ret$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash (\sigma){:}(\mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}')$
res_jtype	::=	
	|	$\Phi \vdash res \equiv res'$
	|	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Leftarrow res$
	|	$h{:}\mathcal{R}$
object_value_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash object\_value \Rightarrow \texttt{obj}\ \beta$
pval_jtype	::=	
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta$

$$
\begin{array}{lll}
\mathit{spine\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret} \\
 & & \\
\mathit{pexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{comp\_pattern\_jtype} & ::= & \\
 & \mid & \mathit{pattern}{:}\beta \rightsquigarrow \mathcal{C}\, \texttt{with}\, \mathit{term} \\
 & \mid & \mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}\, \texttt{with}\, \mathit{term} \\
 & & \\
\mathit{res\_pattern\_jtype} & ::= & \\
 & \mid & \mathcal{L}; \Phi \vdash \mathit{res}' = \texttt{strip\_ifs}\, (\mathit{res}) \\
 & \mid & \mathcal{L}; \Phi \vdash \mathit{res}\, \texttt{as}\, \mathit{res\_pattern} \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}' \\
 & \mid & \mathcal{L}; \Phi \vdash \mathit{res\_pattern}{:}\mathit{res} \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}' \\
 & & \\
\mathit{ret\_pattern\_jtype} & ::= & \\
 & \mid & \mathcal{L}; \Phi \vdash \overline{\mathit{ret\_pattern}_i}^{\,i} {:}\mathit{ret} \rightsquigarrow \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}' \\
 & & \\
\mathit{tpval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{action\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{memop\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_op} \Rightarrow \mathit{ret}
\end{array}
$$

$$
\begin{array}{lll}
\mathit{tval\_jtype} & ::= & \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{seq\_expr\_jtype} & ::= & \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{is\_expr\_jtype} & ::= & \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{texpr\_jtype} & ::= & \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{subs\_jtype} & ::= & \\
 & | & \mathit{pattern} = \mathit{pval} \rightsquigarrow \sigma \\
 & | & \mathit{ident\_or\_pattern} = \mathit{pval} \rightsquigarrow \sigma \\
 & | & \mathit{res\_pattern} = \mathit{res\_term} \rightsquigarrow \sigma \\
 & | & \overline{\mathit{ret\_pattern}_i = \mathit{spine\_elem}_i}^{\,i} \rightsquigarrow \sigma \\
 & | & \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret} \\
 & & \\
\mathit{pure\_opsem\_jtype} & ::= & \\
 & | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{pexpr}' \rangle \\
 & | & \langle \mathit{pexpr} \rangle \longrightarrow \langle \mathit{tpexpr}{:}(y{:}\beta.\, \mathit{term}) \rangle \\
 & | & \langle \mathit{tpexpr} \rangle \longrightarrow \langle \mathit{tpexpr}' \rangle \\
 & & \\
\mathit{opsem\_jtype} & ::= & \\
 & | & \langle h; \mathit{seq\_expr} \rangle \longrightarrow \langle h'; \mathit{texpr}{:}\mathit{ret} \rangle \\
 & | & \langle h; \mathit{seq\_texpr} \rangle \longrightarrow \langle h'; \mathit{texpr} \rangle \\
 & | & \langle h; \mathit{mem\_op} \rangle \longrightarrow \langle h'; \mathit{tval} \rangle
\end{array}
$$

$$
\begin{array}{ll}
| & \langle h; mem\_action \rangle \longrightarrow \langle h'; tval \rangle \\
| & \langle h; is\_expr \rangle \longrightarrow \langle h'; is\_expr' \rangle \\
| & \langle h; is\_texpr \rangle \longrightarrow \langle h'; texpr \rangle \\
| & \langle h; texpr \rangle \longrightarrow \langle h'; texpr' \rangle
\end{array}
$$

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

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

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

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

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

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

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

$$\frac{}{\cdot;\cdot;\cdot;\cdot \sqsubseteq \cdot;\cdot;\cdot;\cdot} \quad \textsc{Weak\_Empty}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}'; \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{C}, x{:}\beta; \mathcal{L}; \Phi; \mathcal{R} \sqsubseteq \mathcal{C}', x{:}\beta; \mathcal{L}'; \Phi'; \mathcal{R}'} \quad \textsc{Weak\_Cons\_Comp}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L},x{:}\beta;\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}',x{:}\beta;\Phi';\mathcal{R}'}\ \textsc{Weak\_Cons\_Log}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L};\Phi,term;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi',term;\mathcal{R}'}\ \textsc{Weak\_Cons\_Phi}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R},x{:}res \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}',x{:}res}\ \textsc{Weak\_Cons\_Res}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}',x{:}\beta;\mathcal{L}';\Phi';\mathcal{R}'}\ \textsc{Weak\_Skip\_Comp}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}',x{:}\beta;\Phi';\mathcal{R}'}\ \textsc{Weak\_Skip\_Log}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}'}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \sqsubseteq \mathcal{C}';\mathcal{L}';\Phi',term;\mathcal{R}'}\ \textsc{Weak\_Skip\_Phi}$$

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash (\cdot){:}(\cdot;\cdot;\cdot;\cdot)}\ \textsc{Ty\_Subs\_Empty}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (pval/x,\sigma){:}(\mathcal{C}',x{:}\beta;\mathcal{L}';\Phi';\mathcal{R}')}\ \textsc{Ty\_Subs\_Cons\_Comp}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma){:}(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (pval/x,\sigma){:}(\mathcal{C}';\mathcal{L}',x{:}\beta;\Phi';\mathcal{R}')}\ \textsc{Ty\_Subs\_Cons\_Log}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \texttt{smt}\,(\Phi \Rightarrow term)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}';\mathcal{L}';\Phi', term;\mathcal{R}')}\ \textsc{Ty\_Subs\_Cons\_Phi}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\sigma):(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}') \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow \sigma(res)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R},\mathcal{R}_1 \vdash (res\_term/x,\sigma):(\mathcal{C}';\mathcal{L}';\Phi';\mathcal{R}', x{:}res)}\ \textsc{Ty\_Subs\_Cons\_Res}$$

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

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

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

$$\frac{\begin{array}{c}\Phi \vdash res_1 \equiv res_1' \\ \Phi \vdash res_2 \equiv res_2'\end{array}}{\Phi \vdash res_1 * res_2 \equiv res_1' * res_2'}\ \textsc{Ty\_Res\_Eq\_SepConj}$$

$$\frac{\Phi \vdash res \equiv res'}{\Phi \vdash \exists\, ident{:}\beta.\ res \equiv \exists\, ident{:}\beta.\ res'}\ \textsc{Ty\_Res\_Eq\_Exists}$$

$$\frac{\begin{array}{l}\texttt{smt}\,(\Phi \Rightarrow (term \to term') \wedge (term' \to term)) \\ \Phi \vdash res \equiv res'\end{array}}{\Phi \vdash term \wedge res \equiv term' \wedge res'}\ \textsc{Ty\_Res\_Eq\_Term}$$

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

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

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

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash \mathit{res\_term}_1 \Leftarrow \mathit{res}_1 \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash \mathit{res\_term}_2 \Leftarrow \mathit{res}_2}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \langle \mathit{res\_term}_1, \mathit{res\_term}_2 \rangle \Leftarrow \mathit{res}_1 * \mathit{res}_2} \quad \textsc{Ty\_Res\_SepConj}$$

$$\frac{\mathtt{smt}\,(\Phi \Rightarrow \mathit{term}) \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{res}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{term} \wedge \mathit{res}} \quad \textsc{Ty\_Res\_Conj}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{pval}/y, \cdot(\mathit{res})}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{pack}\,(\mathit{pval}, \mathit{res\_term}) \Leftarrow \exists\, y{:}\beta.\ \mathit{res}} \quad \textsc{Ty\_Res\_Pack}$$

$$\frac{\begin{array}{l}\alpha \equiv \overline{x_i{:}\beta_i}^{\,i} \mapsto \mathit{res} \in \mathtt{Globals} \\ \overline{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval}_i \Rightarrow \beta_i}^{\,i} \\ \mathcal{L}; \Phi \vdash \mathit{res}' = \mathtt{strip\_ifs}\,(\overline{\mathit{pval}_i/x_i, \cdot}^{\,i}(\mathit{res})) \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{res}'\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{fold}\,(\mathit{res\_term}) \Leftarrow \alpha(\overline{\mathit{pval}_i}^{\,i})} \quad \textsc{Ty\_Res\_Fold}$$

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

$$\frac{}{\cdot{:}\cdot} \quad \textsc{Ty\_Heap\_Emp}$$

$$\frac{\begin{array}{l} h{:}\mathcal{R} \\ \cdot;\cdot;\cdot;\mathcal{R}' \vdash pt \Leftarrow pt \end{array}}{h + \{pt\}{:}\mathcal{R}, \mathcal{R}'} \quad \textsc{Ty\_Heap\_PointsTo}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\texttt{smt}\,(\Phi \Rightarrow term) \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{x_i = spine\_elem_i}^{\,i} :: term \supset arg \gg \sigma; ret} \quad \textsc{Ty\_Spine\_Phi}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boxed{\mathcal{L}; \Phi \vdash res' = \texttt{strip\_ifs}\,(res)}$

$$\frac{}{\mathcal{L}; \Phi \vdash \texttt{emp} = \texttt{strip\_ifs}\,(\texttt{emp})} \quad \textsc{Ty\_Pat\_Res\_StripIfs\_Empty}$$

$$\frac{}{\mathcal{L}; \Phi \vdash pt = \texttt{strip\_ifs}\,(pt)} \quad \textsc{Ty\_Pat\_Res\_StripIfs\_PointsTo}$$

$$\frac{}{\mathcal{L};\Phi \vdash res_1 * res_2 = \texttt{strip\_ifs}\,(res_1 * res_2)}\ \textsc{Ty\_Pat\_Res\_StripIfs\_SepConj}$$

$$\frac{}{\mathcal{L};\Phi \vdash \exists\, x{:}\beta.\ res = \texttt{strip\_ifs}\,(\exists\, x{:}\beta.\ res)}\ \textsc{Ty\_Pat\_Res\_StripIfs\_Exists}$$

$$\frac{}{\mathcal{L};\Phi \vdash term \wedge res = \texttt{strip\_ifs}\,(term \wedge res)}\ \textsc{Ty\_Pat\_Res\_StripIfs\_TermConj}$$

$$\frac{\begin{array}{c}\texttt{smt}\,(\Phi \Rightarrow term) \\ \mathcal{L};\Phi \vdash res_1' = \texttt{strip\_ifs}\,(res_1')\end{array}}{\mathcal{L};\Phi \vdash res_1' = \texttt{strip\_ifs}\,(\texttt{if}\, term\, \texttt{then}\, res_1\, \texttt{else}\, res_2)}\ \textsc{Ty\_Pat\_Res\_StripIfs\_OrdDisj\_True}$$

$$\frac{\begin{array}{c}\texttt{smt}\,(\Phi \Rightarrow \neg\, term) \\ \mathcal{L};\Phi \vdash res_2' = \texttt{strip\_ifs}\,(res_2)\end{array}}{\mathcal{L};\Phi \vdash res_2' = \texttt{strip\_ifs}\,(\texttt{if}\, term\, \texttt{then}\, res_1\, \texttt{else}\, res_2)}\ \textsc{Ty\_Pat\_Res\_StripIfs\_OrdDisj\_False}$$

$$\frac{}{\mathcal{L};\Phi \vdash \alpha(\overline{pval_i}^{\,i}) = \texttt{strip\_ifs}\,(\alpha(\overline{pval_i}^{\,i}))}\ \textsc{Ty\_Pat\_Res\_StripIfs\_PredVar}$$

$\boxed{\mathcal{L};\Phi \vdash res\ \texttt{as}\ res\_pattern \rightsquigarrow \mathcal{L}';\Phi';\mathcal{R}'}$

$$\frac{}{\mathcal{L};\Phi \vdash \texttt{emp}\ \texttt{as}\ \texttt{emp} \rightsquigarrow \cdot;\cdot;\cdot}\ \textsc{Ty\_Pat\_Res\_Match\_Empty}$$

$$\frac{}{\mathcal{L};\Phi \vdash res\ \texttt{as}\ r \rightsquigarrow \cdot;\cdot;\cdot,\, r{:}res}\ \textsc{Ty\_Pat\_Res\_Match\_Var}$$

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

$$\frac{\mathcal{L}; \Phi \vdash \mathit{res\_pattern}{:}\mathit{res} \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{L}; \Phi \vdash \mathit{term} \wedge \mathit{res}\ \texttt{as}\ \mathit{res\_pattern} \rightsquigarrow \mathcal{L}'; \Phi', \mathit{term}; \mathcal{R}'}\ \textsc{Ty\_Pat\_Res\_Match\_Conj}$$

$$\frac{\mathcal{L}; \Phi \vdash \mathit{res\_pattern}{:}x/y, \cdot(\mathit{res}) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}{\mathcal{L}; \Phi \vdash \exists\, y{:}\beta.\ \mathit{res}\ \texttt{as}\ \texttt{pack}\,(x, \mathit{res\_pattern}) \rightsquigarrow \mathcal{L}', x{:}\beta; \Phi'; \mathcal{R}'}\ \textsc{Ty\_Pat\_Res\_Match\_Pack}$$

$$\frac{\begin{array}{c}\alpha \equiv \overline{x_i{:}\beta_i}^{\,i} \mapsto \mathit{res} \in \texttt{Globals} \\ \mathcal{L}; \Phi \vdash \mathit{res\_pattern}{:}\overline{\mathit{pval}_i/x_i, \cdot}^{\,i}(\mathit{res}) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'\end{array}}{\mathcal{L}; \Phi \vdash \alpha(\overline{\mathit{pval}_i}^{\,i})\ \texttt{as}\ \texttt{fold}\,(\mathit{res\_pattern}) \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}\ \textsc{Ty\_Pat\_Res\_Match\_Fold}$$

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

$$\frac{\begin{array}{c}\mathcal{L}; \Phi \vdash \mathit{res}' = \texttt{strip\_ifs}\,(\mathit{res}) \\ \mathcal{L}; \Phi \vdash \mathit{res}'\ \texttt{as}\ \mathit{res\_pattern} \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'\end{array}}{\mathcal{L}; \Phi \vdash \mathit{res\_pattern}{:}\mathit{res} \rightsquigarrow \mathcal{L}'; \Phi'; \mathcal{R}'}\ \textsc{Ty\_Pat\_Res\_Strip\_Ifs}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{done}\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \mathit{pval}/y, \cdot(\mathit{ret})\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{done}\ \mathit{pval},\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \Sigma\, y{:}\beta.\ \mathit{ret}} \quad \textsc{Ty\_TVal\_Comp}$$

$$\frac{\begin{array}{l}\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{done}\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \mathit{pval}/y, \cdot(\mathit{ret})\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{done}\ \mathit{pval},\ \overline{\mathit{spine\_elem}_i}^{\,i} \Leftarrow \exists\, y{:}\beta.\ \mathit{ret}} \quad \textsc{Ty\_TVal\_Log}$$

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

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

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

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

$$\frac{\begin{array}{l}\mathit{ident}{:}\mathit{arg} \equiv \overline{x_i}^{\,i} \mapsto \mathit{texpr} \in \texttt{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \sigma; \mathit{ret}\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \texttt{ccall}\,(\tau, \mathit{ident}, \overline{\mathit{spine\_elem}_i}^{\,i}) \Rightarrow \sigma(\mathit{ret})} \quad \textsc{Ty\_Seq\_E\_CCall}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}pattern_1 = pval_1 \rightsquigarrow \sigma_1 \\ pattern_2 = pval_2 \rightsquigarrow \sigma_2\end{array}}{\texttt{Cons}(pattern_1, pattern_2) = \texttt{Cons}(pval_1, pval_2) \rightsquigarrow \sigma_1, \sigma_2}\ \textsc{Subs\_Decons\_Value\_Cons}$$

$$\frac{\overline{pattern_i = pval_i \rightsquigarrow \sigma_i}^{\,i}}{\texttt{Tuple}(\overline{pattern_i}^{\,i}) = \texttt{Tuple}(\overline{pval_i}^{\,i}) \rightsquigarrow \overline{\sigma_i}^{\,i}}\ \textsc{Subs\_Decons\_Value\_Tuple}$$

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{\rightsquigarrow .} \quad \textsc{Subs\_Decons\_Ret\_Empty}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{\langle h; \texttt{if True then}\ texpr_1\ \texttt{else}\ texpr_2\rangle \longrightarrow \langle h; texpr_1\rangle}\ \textsc{Op_STE_TE_If_True}$$

$$\frac{}{\langle h; \texttt{if False then}\ texpr_1\ \texttt{else}\ texpr_2\rangle \longrightarrow \langle h; texpr_2\rangle}\ \textsc{Op_STE_TE_If_False}$$

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

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

$$\frac{bool\_value \equiv mem\_int_1\ binop_{rel}\ mem\_int_2}{\langle h; mem\_int_1\ \ binop_{rel}\ \ mem\_int_2\rangle \longrightarrow \langle h; \texttt{done}\ bool\_value\rangle}\ \textsc{Op_Memop_TVal_Rel_Binop}$$

$$\frac{mem\_int \equiv \texttt{cast\_ptr\_to\_int}\ mem\_ptr}{\langle h; \texttt{intFromPtr}\,(\tau_1, \tau_2, mem\_ptr)\rangle \longrightarrow \langle h; \texttt{done}\ mem\_int\rangle}\ \textsc{Op_Memop_TVal_IntFromPtr}$$

$$\frac{mem\_ptr \equiv \texttt{cast\_ptr\_to\_int}\ mem\_int}{\langle h; \texttt{ptrFromInt}\,(\tau_1, \tau_2, mem\_int)\rangle \longrightarrow \langle h; \texttt{done}\ mem\_ptr\rangle}\ \textsc{Op_Memop_TVal_PtrFromInt}$$

$$\frac{bool\_value \equiv \texttt{aligned}\,(\tau, mem\_ptr)}{\langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_\}; \texttt{ptrValidForDeref}\,(\tau, mem\_ptr, mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_)\rangle \longrightarrow \langle h + \{mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_\}; \texttt{done}\ bool\_value, mem\_ptr \overset{\checkmark}{\mapsto}_\tau \_\rangle}\ \textsc{Op_Memop_TVal_PtrValid}$$

$$\frac{bool\_value \equiv \texttt{aligned}\,(\tau, mem\_ptr)}{\langle h; \texttt{ptrWellAligned}\,(\tau, mem\_ptr)\rangle \longrightarrow \langle h; \texttt{done}\ bool\_value\rangle}\ \textsc{Op_Memop_TVal_PtrWellAligned}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Definition rules:        209 good    0 bad
Definition rule clauses: 466 good    0 bad
```