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)

| $Sctypes\_t,\ \tau$ | $::=$ | | C type |
| | $\|$ | $\tau *$ | pointer to type τ |

| tag | $::=$ | | OCaml type for struct/union tag |
| | $\|$ | $ident$ | |

$\beta,\ \_$	$::=$		base types
	$\|$	`unit`	unit
	$\|$	`bool`	boolean
	$\|$	`integer`	integer
	$\|$	`real`	rational numbers?
	$\|$	`loc`	location
	$\|$	`array` β	array
	$\|$	`list` β	list
	$\|$	$\overline{\beta_i}^{\,i}$	tuple
	$\|$	`struct` tag	struct
	$\|$	`set` β	set
	$\|$	`opt` (β)	option
	$\|$	$\beta \to \beta'$	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
	|	$\texttt{array}(\overline{loaded\_value_i}^{\,i})$	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 |
| | \| | $\texttt{specified}\,object\_value$ | 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
	|	$\texttt{Nil}\,\beta$	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
| $\texttt{constrained}\,(\,\overline{mem\_iv\_c_i, pval_i}^{\,i}\,)$ constrained value
| $\texttt{error}\,(string, pval)$ 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

| $\texttt{undef}\ UB\_name$ undefined behaviour
| $\texttt{done}\ pval$ pure done

$ident\_opt\_\beta$ ::=   type annotated optional identifier

| $\_{:}\beta$ binders = {}

5

| $ident{:}\beta$ binders $= ident$

$pattern$ ::=

| $ident\_opt\_\beta$ binders $=$ binders($ident\_opt\_\beta$)
| $ctor\_val(\overline{pattern_i}^{\,i})$ binders $=$ binders($\overline{pattern_i}^{\,i}$)

z ::=   OCaml arbitrary-width integer
| i M   literal integer
| mem_int M
| size_of(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($pattern$)

$bool\_op$ ::=
| $\neg\, term$

6

$$\begin{array}{lll}
& | & term_1 = term_2 \\
& | & term_1 \rightarrow term_2 \\
& | & \bigwedge(\overline{term_i}^{\,i}) \\
& | & \bigvee(\overline{term_i}^{\,i}) \\
& | & term_1 \, binop_{bool} \, term_2 \qquad \text{M} \\
& | & \texttt{if} \, term_1 \, \texttt{then} \, term_2 \, \texttt{else} \, term_3
\end{array}$$

$$arith\_op \quad ::=$$

$$\begin{array}{lll}
& | & term_1 + term_2 \\
& | & term_1 - term_2 \\
& | & term_1 \times term_2 \\
& | & term_1 / term_2 \\
& | & term_1 \, \texttt{rem\_t} \, term_2 \\
& | & term_1 \, \texttt{rem\_f} \, term_2 \\
& | & term_1 \, \hat{} \, term_2 \\
& | & term_1 \, binop_{arith} \, term_2 \qquad \text{M}
\end{array}$$

$$cmp\_op \quad ::=$$

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

$$list\_op \quad ::=$$

$$\begin{array}{lll}
& | & \texttt{nil} \\
& | & term_1 :: term_2 \\
& | & \texttt{tl} \, term \\
& | & term^{(int)}
\end{array}$$

$$tuple\_op \quad ::=$$

$$\begin{array}{lll}
& | & (\overline{term_i}^{\,i}) \\
& | & term^{(int)}
\end{array}$$

$pointer\_op$ ::=
| mem_ptr
| $term_1 +_{\mathrm{ptr}} term_2$
| $\mathtt{cast\_int\_to\_ptr}\, term$
| $\mathtt{cast\_ptr\_to\_int}\, term$

$array\_op$ ::=
| $[|\ \overline{term_i}^{\,i}\ |]$
| $term_1[term_2]$

$param\_op$ ::=
| $ident{:}\beta.\ term$
| $term(term_1, .., term_n)$

$struct\_op$ ::=
| $term.member$

ct_pred ::=
| $\mathtt{representable}\,(\tau, term)$
| $\mathtt{aligned}\,(\tau, term)$
| $\mathtt{alignedI}\,(term_1, term_2)$

$term,\ \_$ ::=
| 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 applicatio…
	$\texttt{array\_shift}\,(pval_1, \tau, pval_2)$			pointer array shift
	$\texttt{member\_shift}\,(pval, ident, member)$			pointer struct/union memb…
	$\texttt{not}\,(pval)$			boolean not
	$pval_1 \; binop \; pval_2$			binary operations
	$\texttt{memberof}\,(ident, member, pval)$			C struct/union member ac…
	$name(\overline{pval_i}^{\,i})$			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 $tpexpr_2$	annoted pure let
		bind y_1 in $term_1$	
	$\texttt{if}\,pval\,\texttt{then}\,tpexpr_1\,\texttt{else}\,tpexpr_2$			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 branch |

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
| | `fold`(res_term) fold into recursive res. pred.
| | $\sigma(res\_term)$ M   substitution for resource terms

mem_action ::=   memory actions
| | `create`$(pval, \tau)$
| | `create_readonly`$(pval_1, \tau, pval_2)$
| | `alloc`$(pval_1, pval_2)$
| | `kill`$(m\_kill\_kind, pval, pt)$
| | `store`$(bool, \tau, pval_1, pval_2, mem\_order, pt)$ true means store is locking
| | `load`$(\tau, pval, mem\_order, pt)$
| | `rmw`$(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$
| | `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
	|	$\texttt{ptrWellAligned}\,(\tau, pval)$	
	|	$\texttt{ptrArrayShift}\,(pval_1, \tau, pval_2)$	
	|	$\texttt{memcpy}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{memcmp}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{realloc}\,(pval_1, pval_2, pval_3)$	
	|	$\texttt{va\_start}\,(pval_1, pval_2)$	
	|	$\texttt{va\_copy}\,(pval)$	
	|	$\texttt{va\_arg}\,(pval, \tau)$	
	|	$\texttt{va\_end}\,(pval)$	

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
| | $\mathbf{done}\ spine$ | | end of top-level expression |
| | $\mathbf{undef}\ UB\_name$ | | undefined behaviour |

$res\_pattern$ ::=   resource terms
	$\mathbf{emp}$	binders $= \{\}$	empty heap
	$ident$	binders $= ident$	variable
	$\mathbf{fold}\,(res\_pattern)$	binders $= \{\}$	unfold (recursive) predicate
	$\langle res\_pattern_1, res\_pattern_2 \rangle$	binders $=$ binders$(res\_pattern_1) \cup$ binders$(res\_pattern_2)$	seperating-conjunction pair
	$\mathbf{pack}\,(ident, res\_pattern)$	binders $= ident \cup$ binders$(res\_pattern)$	packing for existentials

$ret\_pattern$ ::=   return pattern
	$\mathbf{comp}\,ident\_or\_pattern$	binders $=$ binders$(ident\_or\_pattern)$	computational variable
	$\mathbf{log}\,ident$	binders $= ident$	logical variable
	$\mathbf{res}\,res\_pattern$	binders $=$ binders$(res\_pattern)$	resource variable

$init,$ ::=   initialisation status
| | $\checkmark$ | | initialised |
| | $\times$ | | 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 disjunction
	$\alpha(\overline{pval_i}^{\,i})$		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) expressions

	$tval$		(effectful) top-level values
	$\texttt{run}\, ident\, \overline{pval_i}^{\,i}$		run from label
	let $ident\_or\_pattern = pexpr$ in $texpr$	bind binders($ident\_or\_pattern$) in $texpr$	pure let
	let $ident\_or\_pattern{:}(y_1{:}\beta_1.\ term_1) = tpexpr$ in $texpr$	bind binders($ident\_or\_pattern$) in $texpr$ bind y_1 in $term_1$	annotated pure let
	let $\overline{ret\_pattern_i}^{\,i} = seq\_expr$ in $texpr$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr$	bind return patterns
	let $\overline{ret\_pattern_i}^{\,i}{:}ret = texpr_1$ in $texpr_2$	bind binders($\overline{ret\_pattern_i}^{\,i}$) in $texpr_2$	annotated bind return patterns
	case $pval$ of $\overline{\mid texpr\_case\_branch_i}^{\,i}$ end		pattern matching
	if $pval$ then $texpr_1$ else $texpr_2$		conditional

13

| | bound $[int](is\_texpr)$ | | limit scope of indet seq behaviour, absent at ru |

| $texpr\_case\_branch$ | ::= | | 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) expressions
	$\texttt{let}\,\texttt{weak}\,\overline{ret\_pattern_i}^{\,i} = is\_expr\,\texttt{in}\,texpr$	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
	is_texpr		indet seq (effectful) expressions
	$\sigma(texpr)$	M	simul-sub σ in $texpr$

arg	::=		argument/function types
	$\Pi\,ident{:}\beta.\,arg$		
	$\forall\,ident{:}\beta.\,arg$		
	$res \multimap arg$		
	$term \supset arg$		
	ret		
	$\sigma(arg)$	M	simul-sub σ in arg

$pure\_arg$	::=		pure argument/function types
	$\Pi\,ident{:}\beta.\,pure\_arg$		
	$term \supset pure\_arg$		
	$pure\_ret$		

pure_ret	::=		pure return types
	|	$\Sigma\, ident{:}\beta.\ pure\_ret$	
	|	$term \wedge pure\_ret$	
	|	$\mathtt{I}$	

$\mathcal{C}$	::=		computational var env
	|	$\cdot$	
	|	$\mathcal{C}, ident{:}\beta$	
	|	$\overline{\mathcal{C}_i}^{\,i}$	

$\mathcal{L}$	::=		logical var env
	|	$\cdot$	
	|	$\overline{\mathcal{L}_i}^{\,i}$	
	|	$\mathcal{L}, ident{:}\beta$	

Φ	::=		constraints env
	|	$\cdot$	
	|	$\Phi, term$	
	|	$\overline{\Phi_i}^{\,i}$	

$\mathcal{R}$	::=		resources env
	|	$\cdot$	
	|	$\mathcal{R}, ident{:}res$	
	|	$\overline{\mathcal{R}_i}^{\,i}$	

$\sigma,\ \psi$	::=		substitutions
	|	$\cdot$	
	|	$spine\_elem/ident, \sigma$	
	|	$term/ident, \sigma$	
	|	$\overline{\sigma_i}^{\,i}$	
	|	$\sigma(\psi)$ 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

$opsem$ ::=

| $\forall\, i < j.\ \texttt{not}\,(pattern_i = pval \rightsquigarrow \sigma_i)$
| $\texttt{fresh}\,(mem\_ptr)$
| $term$
| $pval{:}\beta$

$formula$ ::=

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

$heap,\ h,\ f$ ::=   heaps

| $\cdot$
| $h + \{points\_to\}$
| $h + f$ [O]   convenient for the soundness proof

$lemma\_jtype$ ::=

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

res_jtype $::=$

| $\Phi \vdash res \equiv res'$

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

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

$object\_value\_jtype$ $::=$

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

$pval\_jtype$ $::=$

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

$spine\_jtype$ $::=$

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

$pexpr\_jtype$ $::=$

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

$comp\_pattern\_jtype$ $::=$

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

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

$res\_pattern\_jtype$ $::=$

| $\Phi \vdash res' = \texttt{strip\_ifs}\,(res)$

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

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

$ret\_pattern\_jtype$ $::=$

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

$tpval\_jtype$ $::=$

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

$tpexpr\_jtype$ ::=

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

$action\_jtype$ ::=

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

$memop\_jtype$ ::=

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

$tval\_jtype$ ::=

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

seq_expr_jtype ::=

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

is_expr_jtype ::=

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

$texpr\_jtype$ ::=

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

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

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

$subs\_jtype$ ::=

         | $pattern = pval \rightsquigarrow \sigma$

         | $ident\_or\_pattern = pval \rightsquigarrow \sigma$

         | $res\_pattern = res\_term \rightsquigarrow \sigma$

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

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

$pure\_opsem\_jtype$ ::=

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{\Phi \vdash \mathtt{emp} \equiv \mathtt{emp}} \quad \text{Ty\_Res\_Eq\_Emp}$$

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

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

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

$$\dfrac{\begin{array}{c}\mathtt{smt}\,(\Phi \Rightarrow (term \to term') \wedge (term' \to term)) \\ \Phi \vdash res \,\equiv\, res'\end{array}}{\Phi \vdash term \wedge res \,\equiv\, term' \wedge res'}\quad\text{Ty\_Res\_Eq\_Term}$$

$$\dfrac{\begin{array}{c}\mathtt{smt}\,(\Phi \Rightarrow (term_1 \to term_2) \wedge (term_2 \to term_1)) \\ \Phi \vdash res_{11} \,\equiv\, res_{21} \\ \Phi \vdash res_{21} \,\equiv\, res_{22}\end{array}}{\Phi \vdash \mathtt{if}\ term_1\ \mathtt{then}\ res_{11}\ \mathtt{else}\ res_{12} \,\equiv\, \mathtt{if}\ term_2\ \mathtt{then}\ res_{21}\ \mathtt{else}\ res_{22}}\quad\text{Ty\_Res\_Eq\_OrdDisj}$$

$$\dfrac{}{\Phi \vdash \alpha(\overline{pval_i}^{\,i}) \,\equiv\, \alpha(\overline{pval_i}^{\,i})}\quad\text{Ty\_Res\_Eq\_Pred}$$

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

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

$$\dfrac{\begin{array}{c}\Phi \vdash points\_to \,\equiv\, points\_to' \\ \Phi \vdash points\_to' \,\equiv\, points\_to''\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,\_:points\_to \vdash points\_to' \Leftarrow points\_to''}\quad\text{Ty\_Res\_PointsTo}$$

$$\dfrac{\begin{array}{c}\Phi \vdash res_1' = \mathtt{strip\_ifs}\,(res_1) \\ \Phi \vdash res_2' = \mathtt{strip\_ifs}\,(res_2) \\ \Phi \vdash res_1' \,\equiv\, res_2'\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,r{:}res_1 \vdash r \Leftarrow res_2}\quad\text{Ty\_Res\_Var}$$

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

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

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

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

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

$$\dfrac{}{\because} \quad \textsc{Ty\_Heap\_Emp}$$

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

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

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

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

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

$$\frac{\begin{array}{c}\mathtt{struct}\,tag\ \&\,\overline{member_i{:}\tau_i}^{\ i} \in \mathtt{Globals}\\[2pt] \overline{\mathcal{C};\mathcal{L};\Phi \vdash mem\_val_i \Rightarrow \mathtt{mem}\,\beta_{\tau_i}}^{\ i}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash (\,\mathtt{struct}\,tag)\{\,\overline{.\,member_i{:}\tau_i = mem\_val_i}^{\ i}\,\} \Rightarrow \mathtt{obj\,struct}\,tag}\ \ \text{Ty\_Pval\_Obj\_Struct}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{bool} \qquad \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{bool}}{\mathcal{C};\mathcal{L};\Phi \vdash pval_1\ binop_{bool}\ pval_2 \Rightarrow y{:}\texttt{bool}.\ y = (pval_1\ binop_{bool}\ pval_2)} \quad \text{Ty\_PE\_Bool\_Binop}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boxed{\Phi \vdash res' = \mathtt{strip\_ifs}\,(res)}$

$$\frac{}{\Phi \vdash \mathtt{emp} = \mathtt{strip\_ifs}\,(\mathtt{emp})} \quad \text{Ty\_Pat\_Res\_StripIfs\_Empty}$$

$$\frac{}{\Phi \vdash pt = \mathtt{strip\_ifs}\,(pt)} \quad \text{Ty\_Pat\_Res\_StripIfs\_PointsTo}$$

$$\frac{}{\Phi \vdash res_1 * res_2 = \mathtt{strip\_ifs}\,(res_1 * res_2)} \quad \text{Ty\_Pat\_Res\_StripIfs\_SepConj}$$

$$\frac{}{\Phi \vdash \exists\, x{:}\beta.\, res = \mathtt{strip\_ifs}\,(\exists\, x{:}\beta.\, res)} \quad \text{Ty\_Pat\_Res\_StripIfs\_Exists}$$

$$\frac{}{\Phi \vdash term \wedge res = \mathtt{strip\_ifs}\,(term \wedge res)} \quad \text{Ty\_Pat\_Res\_StripIfs\_TermConj}$$

$$\frac{\begin{array}{c} \mathtt{smt}\,(\Phi \Rightarrow term) \\ \Phi \vdash res_1' = \mathtt{strip\_ifs}\,(res_1') \end{array}}{\Phi \vdash res_1' = \mathtt{strip\_ifs}\,(\mathtt{if}\,term\,\mathtt{then}\,res_1\,\mathtt{else}\,res_2)} \quad \text{Ty\_Pat\_Res\_StripIfs\_True}$$

$$\frac{\begin{array}{c} \mathtt{smt}\,(\Phi \Rightarrow \neg\,term) \\ \Phi \vdash res_2' = \mathtt{strip\_ifs}\,(res_2) \end{array}}{\Phi \vdash res_2' = \mathtt{strip\_ifs}\,(\mathtt{if}\,term\,\mathtt{then}\,res_1\,\mathtt{else}\,res_2)} \quad \text{Ty\_Pat\_Res\_StripIfs\_False}$$

$$\frac{}{\Phi \vdash \mathtt{if}\ term\ \mathtt{then}\ res_1\ \mathtt{else}\ res_2 = \mathtt{strip\_ifs}\,(\mathtt{if}\ term\ \mathtt{then}\ res_1\ \mathtt{else}\ res_2)} \quad \text{Ty\_Pat\_Res\_StripIfs\_UnderDet}$$

$$\frac{}{\Phi \vdash \alpha(\overline{pval_i}^{\,i}) = \mathtt{strip\_ifs}\,(\alpha(\overline{pval_i}^{\,i}))} \quad \text{Ty\_Pat\_Res\_StripIfs\_Pred}$$

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

$$\frac{}{\Phi \vdash \mathtt{emp}\ \mathtt{as}\ \mathtt{emp} \rightsquigarrow \cdot; \cdot; \cdot} \quad \text{Ty\_Pat\_Res\_Match\_Empty}$$

$$\frac{}{\Phi \vdash res\ \mathtt{as}\ r \rightsquigarrow \cdot; \cdot; \cdot, r{:}res} \quad \text{Ty\_Pat\_Res\_Match\_Var}$$

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

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

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

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

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

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

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

$$\frac{}{\Phi \vdash \,{:}\texttt{I} \rightsquigarrow \cdot;\cdot;\cdot;\cdot}\;\; \text{Ty\_Pat\_Ret\_Empty}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c} \textit{name}{:}\textit{arg} \equiv \overline{x_i}^{\,i} \mapsto \textit{texpr} \ \in \mathtt{Globals} \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{x_i = \textit{spine\_elem}_i}^{\,i} :: \textit{arg} \gg \sigma; \textit{ret} \end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{pcall} \ (\textit{name}, \overline{\textit{spine\_elem}_i}^{\,i}) \Rightarrow \sigma(\textit{ret})} \quad \text{TY\_SEQ\_E\_PROC}$$

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_op} \Rightarrow \textit{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{memop} \ (\textit{mem\_op}) \Rightarrow \textit{ret}} \quad \text{TY\_IS\_E\_MEMOP}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_action} \Rightarrow \textit{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_action} \Rightarrow \textit{ret}} \quad \text{TY\_IS\_E\_ACTION}$$

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \textit{mem\_action} \Rightarrow \textit{ret}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathtt{neg} \ \textit{mem\_action} \Rightarrow \textit{ret}} \quad \text{TY\_IS\_E\_NEG\_ACTION}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{ident = res\_term \leadsto res\_term/ident, \cdot} \quad \text{S\textsc{ubs\_Decons\_Res\_Var}}$$

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

$$\frac{res\_pattern = res\_term \leadsto \sigma}{\texttt{pack}\,(ident, res\_pattern) = \texttt{pack}\,(pval, res\_term) \leadsto pval/ident, \sigma} \quad \text{S\textsc{ubs\_Decons\_Res\_Pack}}$$

$$\frac{res\_pattern = res\_term \leadsto \sigma}{\texttt{fold}\,(res\_pattern) = res\_term \leadsto \sigma} \quad \text{S\textsc{ubs\_Decons\_Res\_Fold}}$$

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

$$\frac{}{\leadsto \cdot} \quad \text{S\textsc{ubs\_Decons\_Ret\_Empty}}$$

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

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

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

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

$$\frac{}{:: ret \gg \cdot; ret} \quad \text{S{\sc UBS\_DECONS\_ARG\_EMPTY}}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine\_elem_i}^{\,i} :: \Pi\, x{:}\beta.\; arg \gg pval/x, \sigma; ret} \quad \text{S{\sc UBS\_DECONS\_ARG\_COMP}}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = pval, \overline{x_i = spine\_elem_i}^{\,i} :: \forall\, x{:}\beta.\; arg \gg pval/x, \sigma; ret} \quad \text{S{\sc UBS\_DECONS\_ARG\_LOG}}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{x = res\_term, \overline{x_i = spine\_elem_i}^{\,i} :: res \multimap arg \gg res\_term/x, \sigma; ret} \quad \text{S{\sc UBS\_DECONS\_ARG\_RES}}$$

$$\frac{\overline{x_i = spine\_elem_i}^{\,i} :: arg \gg \sigma; ret}{\overline{x_i = spine\_elem_i}^{\,i} :: term \supset arg \gg \sigma; ret} \quad \text{S{\sc UBS\_DECONS\_ARG\_PHI}}$$

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

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

$$\frac{mem\_ptr' \equiv mem\_ptr +_{\text{ptr}} \text{offset\_of}_{tag}(member)}{\langle \texttt{member\_shift}\,(mem\_ptr, tag, member) \rangle \longrightarrow \langle mem\_ptr' \rangle} \quad \text{O{\sc P\_PE\_PE\_MEMBERSHIFT}}$$

$$\frac{}{\langle \texttt{not}\,(\texttt{True}) \rangle \longrightarrow \langle \texttt{False} \rangle} \quad \text{O{\sc P\_PE\_PE\_NOT\_TRUE}}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{\langle \mathtt{if}\ \mathsf{True}\ \mathtt{then}\ tpexpr_1\ \mathtt{else}\ tpexpr_2 \rangle \longrightarrow \langle tpexpr_1 \rangle}\quad \text{Op\_TPE\_TPE\_If\_True}$$

$$\frac{}{\langle \mathtt{if}\ \mathsf{False}\ \mathtt{then}\ tpexpr_1\ \mathtt{else}\ tpexpr_2 \rangle \longrightarrow \langle tpexpr_2 \rangle}\quad \text{Op\_TPE\_TPE\_If\_False}$$

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

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

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

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

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

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

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle h; \texttt{let}\,ident\_or\_pattern = pval\,\texttt{in}\,texpr\rangle \longrightarrow \langle h; \sigma(texpr)\rangle} \quad \text{OP\_STE\_TE\_LETP\_SUB}$$

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

$$\frac{\langle pexpr\rangle \longrightarrow \langle tpexpr{:}(y{:}\beta.\ term)\rangle}{\langle h; \texttt{let}\,ident\_or\_pattern = pexpr\,\texttt{in}\,texpr\rangle \longrightarrow \langle h; \texttt{let}\,ident\_or\_pattern{:}(y{:}\beta.\ term) = tpexpr\,\texttt{in}\,texpr\rangle} \quad \text{OP\_STE\_TE\_LETP\_LETTP}$$

$$\frac{ident\_or\_pattern = pval \rightsquigarrow \sigma}{\langle h; \texttt{let } ident\_or\_pattern{:}(y{:}\beta.\ term) = \texttt{done } pval \texttt{ in } texpr\rangle \longrightarrow \langle h; \sigma(texpr)\rangle} \quad \text{OP\_STE\_TE\_LETTP\_SUB}$$

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

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

$$\frac{\langle h; seq\_expr\rangle \longrightarrow \langle h; texpr_1{:}ret\rangle}{\langle h; \texttt{let } \overline{ret\_pattern_i}^{\,i} = seq\_expr \texttt{ in } texpr_2\rangle \longrightarrow \langle h; \texttt{let } \overline{ret\_pattern_i}^{\,i}{:}ret = texpr_1 \texttt{ in } texpr_2\rangle} \quad \text{OP\_STE\_TE\_LET\_LETT}$$

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

$$\frac{}{\langle h; \texttt{if True then } texpr_1 \texttt{ else } texpr_2\rangle \longrightarrow \langle h; texpr_1\rangle} \quad \text{OP\_STE\_TE\_IF\_TRUE}$$

$$\frac{}{\langle h; \texttt{if False then } texpr_1 \texttt{ else } texpr_2\rangle \longrightarrow \langle h; texpr_2\rangle} \quad \text{OP\_STE\_TE\_IF\_FALSE}$$

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

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

$$\frac{bool\_value \equiv mem\_int_1\ binop_{rel}\ mem\_int_2}{\langle h; mem\_int_1\ binop_{rel}\ mem\_int_2\rangle \longrightarrow \langle h; \texttt{done } bool\_value\rangle} \quad \text{OP\_MEMOP\_TVAL\_REL\_BINOP}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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