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

Sctypes_t, τ	::=			C type
	|	τ*		pointer to type τ
tag	::=			OCaml type for struct/union tag
	|	*ident*		
β, _	::=			base types
	|	`unit`		unit
	|	`bool`		boolean
	|	`integer`		integer
	|	`real`		rational numbers?
	|	`loc`		location
	|	`array` β		array
	|	`list` β		list
	|	$\overline{\beta_i}^{\,i}$		tuple
	|	`struct` *tag*		struct
	|	`set` β		set
	|	`opt` (β)		option
	|	$\beta \rightarrow \beta'$		parameter types
	|	β_τ	M	of a C type
binop	::=			binary operators
	|	`+`		addition
	|	`-`		subtraction
	|	`*`		multiplication
	|	`/`		division
	|	`rem_t`		modulus
	|	`rem_f`		remainder
	|	`^`		exponentiation
	|	`=`		equality, defined both for integer and C types

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

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

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

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

object_value	::=		C object values (inhabitants of object types), which can be read/stored
	|	*mem_int*	integer value
	|	*mem_ptr*	pointer value

	$\mid$	$\texttt{array}\,(\overline{loaded\_value_i}^{\,i})$	C array value
	$\mid$	$(\texttt{struct}\,ident)\{\,\overline{.member_i{:}\tau_i = mem\_val_i}^{\,i}\,\}$	C struct value
	$\mid$	$(\texttt{union}\,ident)\{\,.member = mem\_val\}$	C union value
loaded_value	::=		potentially unspecified C object values
	$\mid$	$\texttt{specified}\,object\_value$	specified loaded value
value	::=		Core values
	$\mid$	*object_value*	C object value
	$\mid$	*loaded_value*	loaded C object value
	$\mid$	`Unit`	unit
	$\mid$	`True`	boolean true
	$\mid$	`False`	boolean false
	$\mid$	$\beta[\overline{value_i}^{\,i}]$	list
	$\mid$	$(\overline{value_i}^{\,i})$	tuple
bool_value	::=		Core booleans
	$\mid$	`True`	boolean true
	$\mid$	`False`	boolean false
ctor_val	::=		data constructors
	$\mid$	$\texttt{Nil}\,\beta$	empty list
	$\mid$	`Cons`	list cons
	$\mid$	`Tuple`	tuple
	$\mid$	`Array`	C array
	$\mid$	`Specified`	non-unspecified loaded value
ctor_expr	::=		data constructors
	$\mid$	`Ivmax`	max integer value
	$\mid$	`Ivmin`	min integer value

	$\mid$	`Ivsizeof`	sizeof value
	$\mid$	`Ivalignof`	alignof value
	$\mid$	`IvCOMPL`	bitwise complement
	$\mid$	`IvAND`	bitwise AND
	$\mid$	`IvOR`	bitwise OR
	$\mid$	`IvXOR`	bitwise XOR
	$\mid$	`Fvfromint`	cast integer to floating value
	$\mid$	`Ivfromfloat`	cast floating to integer value
name	::=		
	$\mid$	*ident*	Core identifier
	$\mid$	*impl_const*	implementation-defined constant
pval	::=		pure values
	$\mid$	*ident*	Core identifier
	$\mid$	*impl_const*	implementation-defined constant
	$\mid$	*value*	Core values
	$\mid$	`constrained` $(\overline{mem\_iv\_c_i, pval_i}^{\,i})$	constrained value
	$\mid$	`error` $(string, pval)$	impl-defined static error
	$\mid$	$ctor\_val(\overline{pval_i}^{\,i})$	data constructor application
	$\mid$	$(\texttt{struct}\, ident)\{\,\overline{.member_i = pval_i}^{\,i}\,\}$	C struct expression
	$\mid$	$(\texttt{union}\, ident)\{\,.member = pval\}$	C union expression
pexpr	::=		pure expressions
	$\mid$	*pval*	pure values
	$\mid$	$ctor\_expr(\overline{pval_i}^{\,i})$	data constructor application
	$\mid$	`array_shift` $(pval_1, \tau, pval_2)$	pointer array shift
	$\mid$	`member_shift` $(pval, ident, member)$	pointer struct/union member shift
	$\mid$	`not` $(pval)$	boolean not
	$\mid$	$pval_1\ binop\ pval_2$	binary operations

	$\mid$	$\texttt{memberof}\,(ident, member, pval)$	C struct/union member access
	$\mid$	$name(\overline{pval_i}^{\,i})$	pure function call
	$\mid$	$\texttt{assert\_undef}\,(pval,\ UB\_name)$	
	$\mid$	$\texttt{bool\_to\_integer}\,(pval)$	
	$\mid$	$\texttt{conv\_int}\,(\tau, pval)$	
	$\mid$	$\texttt{wrapI}\,(\tau, pval)$	
$tpval$	::=		top-level pure values
	$\mid$	$\texttt{undef}\ UB\_name$	undefined behaviour
	$\mid$	$\texttt{done}\,pval$	pure done
$ident\_opt\_\beta$	::=		type annotated optional identifier
	$\mid$	$\_{:}\beta$ binders = {}	
	$\mid$	$ident{:}\beta$ binders = $ident$	
$pattern$	::=		
	$\mid$	$ident\_opt\_\beta$ binders = binders($ident\_opt\_\beta$)	
	$\mid$	$ctor\_val(\overline{pattern_i}^{\,i})$ binders = binders($\overline{pattern_i}^{\,i}$)	
$ident\_or\_pattern$	::=		
	$\mid$	$ident$ binders = $ident$	
	$\mid$	$pattern$ binders = binders($pattern$)	
$tpexpr$	::=		top-level pure expressions
	$\mid$	$tpval$	top-level pure values
	$\mid$	$\texttt{case}\,pval\,\texttt{of}\,\overline{\mid\ tpexpr\_case\_branch_i}^{\,i}\,\texttt{end}$	pattern matching
	$\mid$	$\texttt{let}\,ident\_or\_pattern = pexpr\,\texttt{in}\,tpexpr$ bind binders($ident\_or\_pattern$) in $tpexpr$	pure let
	$\mid$	$\texttt{if}\,pval\,\texttt{then}\,tpexpr_1\,\texttt{else}\,tpexpr_2$	pure if
	$\mid$	$[\mathcal{C}/\mathcal{C}']tpexpr$ M	simul-sub all vars in $\mathcal{C}$ for all vars in $\mathcal{C}'$ in $tpexp$

tpexpr_case_branch	::=			pure top-level case expression
	|	*pattern* ⇒ *tpexpr*	bind binders(*pattern*) in *tpexpr*	top-level case expression br
m_kill_kind	::=			
	|	`dynamic`		
	|	`static` τ		
bool, _	::=			OCaml booleans
	|	`true`		
	|	`false`		
int, _	::=			OCaml fixed-width integer
	|	i		literal integer
mem_action	::=			memory actions
	|	`create` $(pval, \tau)$		
	|	`create_readonly` $(pval_1, \tau, pval_2)$		
	|	`alloc` $(pval_1, pval_2)$		
	|	`kill` $(m\_kill\_kind, pval)$		
	|	`store` $(bool, \tau, pval_1, pval_2, mem\_order)$		true means store is locking
	|	`load` $(\tau, pval, mem\_order)$		
	|	`rmw` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`fence` (mem_order)		
	|	`cmp_exch_strong` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`cmp_exch_weak` $(\tau, pval_1, pval_2, pval_3, mem\_order_1, mem\_order_2)$		
	|	`linux_fence` $(linux\_mem\_order)$		
	|	`linux_load` $(\tau, pval, linux\_mem\_order)$		
	|	`linux_store` $(\tau, pval_1, pval_2, linux\_mem\_order)$		
	|	`linux_rmw` $(\tau, pval_1, pval_2, linux\_mem\_order)$		

polarity	::=		polarities for memory actions
	|	`Pos`	sequenced by `let weak` and `let strong`
	|	`Neg`	only sequenced by `let strong`
pol_mem_action	::=		memory actions with polarity
	|	*polarity* *mem_action*	
mem_op	::=		operations involving the memory state
	|	$pval_1 \equiv pval_2$	pointer equality comparison
	|	$pval_1 \neq pval_2$	pointer inequality comparison
	|	$pval_1 < pval_2$	pointer less-than comparison
	|	$pval_1 > pval_2$	pointer greater-than comparison
	|	$pval_1 \leq pval_2$	pointer less-than comparison
	|	$pval_1 \geq pval_2$	pointer greater-than comparison
	|	$pval_1 -_\tau pval_2$	pointer subtraction
	|	`intFromPtr` $(\tau_1, \tau_2, pval)$	cast of pointer value to integer value
	|	`ptrFromInt` $(\tau_1, \tau_2, pval)$	cast of integer value to pointer value
	|	`ptrValidForDeref` $(\tau, pval)$	dereferencing validity predicate
	|	`ptrWellAligned` $(\tau, pval)$	
	|	`ptrArrayShift` $(pval_1, \tau, pval_2)$	
	|	`memcpy` $(pval_1, pval_2, pval_3)$	
	|	`memcmp` $(pval_1, pval_2, pval_3)$	
	|	`realloc` $(pval_1, pval_2, pval_3)$	
	|	`va_start` $(pval_1, pval_2)$	
	|	`va_copy` $(pval)$	
	|	`va_arg` $(pval, \tau)$	
	|	`va_end` $(pval)$	
res_term	::=		resource terms
	|	`emp`	empty heap

	$\mid$	$\texttt{pt}$	single-cell heap
	$\mid$	$ident$	variable
	$\mid$	$\langle res\_term_1, res\_term_2 \rangle$	seperating-conjunction pair
	$\mid$	$\texttt{pack}\,(pval, res\_term_2)$	packing for existentials
$spine\_elem$	::=		spine element
	$\mid$	$pval$	pure value
	$\mid$	$term$	logical value
	$\mid$	res_term	resource value
$tval$	::=		(effectful) top-level values
	$\mid$	$\texttt{done}\,\overline{spine\_elem_i}^{\,i}$	end of top-level expression
	$\mid$	$\texttt{undef}\ UB\_name$	undefined behaviour

$res\_pattern$	::=			resource terms
	$\mid$	$\texttt{emp}$	$\text{binders} = \{\}$	empty heap
	$\mid$	$\texttt{pt}$	$\text{binders} = \{\}$	single-cell heap
	$\mid$	$ident$	$\text{binders} = ident$	variable
	$\mid$	$\langle res\_pattern_1, res\_pattern_2 \rangle$	$\text{binders} = \text{binders}(res\_pattern_1) \cup \text{binders}(res\_pattern_2)$	seperating-conjunction pair
	$\mid$	$\texttt{pack}\,(ident, res\_pattern)$	$\text{binders} = ident \cup \text{binders}(res\_pattern)$	packing for existentials
$ret\_pattern$	::=			return pattern
	$\mid$	$\texttt{comp}\,ident\_or\_pattern$	$\text{binders} = \text{binders}(ident\_or\_pattern)$	computational variable
	$\mid$	$\texttt{log}\,ident$	$\text{binders} = ident$	logical variable
	$\mid$	$\texttt{res}\,res\_pattern$	$\text{binders} = \text{binders}(res\_pattern)$	resource variable

$bool\_op$	::=	
	$\mid$	$\neg\, term$
	$\mid$	$term_1 = term_2$
	$\mid$	$\bigwedge(\overline{term_i}^{\,i})$

	$\bigvee(\overline{term_i}^{\,i})$		
	$term_1\ binop_{bool}\ term_2$	M	
	$\texttt{if}\ term_1\ \texttt{then}\ term_2\ \texttt{else}\ term_3$		
arith_op	::=		
	$term_1 + term_2$		
	$term_1 - term_2$		
	$term_1 \times term_2$		
	$term_1/term_2$		
	$term_1\ \texttt{rem\_t}\ term_2$		
	$term_1\ \texttt{rem\_f}\ term_2$		
	$term_1\ \texttt{\^{}}\ term_2$		
	$term_1\ binop_{arith}\ term_2$	M	
cmp_op	::=		
	$term_1 < term_2$		less than
	$term_1 \leq term_2$		less than or equal
	$term_1\ binop_{rel}\ term_2$	M	
list_op	::=		
	$\texttt{nil}$		
	$\texttt{tl}\ term$		
	$term^{(int)}$		
tuple_op	::=		
	$(\overline{term_i}^{\,i})$		
	$term^{(int)}$		
pointer_op	::=		
	mem_ptr		

$$
\begin{array}{lll}
 & | & term_1 +_{\mathrm{ptr}} term_2 \\
 & | & \texttt{cast\_int\_to\_ptr}\, term \\
 & | & \texttt{cast\_ptr\_to\_int}\, term \\
 & & \\
\mathit{option\_op} & ::= & \\
 & | & \texttt{none}\, \beta \\
 & | & \texttt{some}\, term \\
 & & \\
\mathit{array\_op} & ::= & \\
 & | & term_1[term_2] \\
 & & \\
\mathit{param\_op} & ::= & \\
 & | & ident{:}\beta.\, term \\
 & | & term(term_1, \ldots, term_n) \\
 & & \\
\mathit{struct\_op} & ::= & \\
 & | & term.member \\
 & & \\
\mathit{ct\_pred} & ::= & \\
 & | & \texttt{representable}\, (\tau, term) \\
 & | & \texttt{aligned}\, (\tau, term) \\
 & | & \texttt{alignedI}\, (term_1, term_2) \\
 & & \\
term,\ \_ & ::= & \\
 & | & lit \\
 & | & \mathit{arith\_op} \\
 & | & \mathit{bool\_op} \\
 & | & \mathit{cmp\_op} \\
 & | & \mathit{tuple\_op} \\
 & | & \mathit{struct\_op}
\end{array}
$$

	$pointer\_op$		
	$list\_op$		
	$array\_op$		
	ct_pred		
	$option\_op$		
	$param\_op$		
	$(term)$	S	parentheses
	$[term_1/ident]term_2$	M	substitute $term_1$ for $ident$ in $term_2$
	$pval$	M	only the ones which can be embeded into the SMT value grammar, so no array literals

$init,$	::=		initialisation status
	✓		initialised
	×		uninitalised

$points\_to$	::=	points-to separation logic predicate
	$term_1 \mathbb{Q}^{init}_{\mapsto_\tau} term_2$	

$resource$	::=		resources
	`emp`		empty heap
	$points\_to$		points-top heap pred.
	$resource_1 \star resource_2$		seperating conjunction
	$\exists\, ident{:}\beta.\ resource$		existential
	$term \wedge resource$		logical conjuction
	$\langle resource \rangle$	S	parentheses
	$[pval/ident]resource$	M	substitute $pval$ for $ident$ in $resource$

$ret,\ \_$	::=	return types
	$\Sigma\, ident{:}\beta.\ ret$	
	$\exists\, ident{:}\beta.\ ret$	
	$resource \star ret$	

	$term \wedge ret$		
	$\texttt{I}$		
	$[spine\_elem/ident]ret$	M	
	$rets$	M	concatenation of return types

$rets$	::=			concatenation of return types
	|	$\overline{ret_i}^{\,i}$		

seq_expr	::=			sequential (effectful) expressions
	|	$pexpr$		pure expressions
	|	$\texttt{ccall}\,(\tau, pval, \overline{spine\_elem_i}^{\,i})$		C function call
	|	$\texttt{pcall}\,(name, \overline{spine\_elem_i}^{\,i})$		procedure call

seq_texpr	::=			sequential top-level (effectful) expressions
	|	$tval$		(effectful) top-level values
	|	$\texttt{run}\,ident\,\overline{pval_i}^{\,i}$		run from label
	|	$\texttt{let}\,\overline{ret\_pattern_i}^{\,i} = seq\_expr\,\texttt{in}\,texpr$	$\mathsf{bind}\ \mathrm{binders}(\overline{ret\_pattern_i}^{\,i})\ \mathsf{in}\ texpr$	bind return patterns
	|	$\texttt{let}\,\overline{ret\_pattern_i}^{\,i} : ret = texpr_1\,\texttt{in}\,texpr_2$	$\mathsf{bind}\ \mathrm{binders}(\overline{ret\_pattern_i}^{\,i})\ \mathsf{in}\ texpr_2$	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, absent at runti

$texpr\_case\_branch$	::=			top-level case expression branch
	|	$pattern \Rightarrow texpr$	$\mathsf{bind}\ \mathrm{binders}(pattern)\ \mathsf{in}\ texpr$	top-level case expression branch

is_expr	::=			indet seq (effectful) expressions
	|	$\texttt{memop}\,(mem\_op)$		pointer op involving memory
	|	pol_mem_action		memory action
	|	$\texttt{unseq}\,(\overline{texpr_i : ret_i}^{\,i})$		unsequenced expressions

is_texpr	::=			indet seq top-level (effectful) expressions
	|	$\texttt{let}\ \texttt{weak}\ \overline{ret\_pattern_i}^{\,i} = is\_expr\ \texttt{in}\ texpr$	$\mathsf{bind}\ \mathrm{binders}(\overline{ret\_pattern_i}^{\,i})\ \mathsf{in}\ texpr$	weak sequencing
	|	$\texttt{let}\ \texttt{strong}\ \overline{ret\_pattern_i}^{\,i} = is\_expr\ \texttt{in}\ texpr$	$\mathsf{bind}\ \mathrm{binders}(\overline{ret\_pattern_i}^{\,i})\ \mathsf{in}\ texpr$	strong sequencing
texpr	::=			top-level (effectful) expressions
	|	seq_texpr		sequential (effectful) expressions
	|	is_texpr		indet seq (effectful) expressions
	|	$[\mathcal{C}/\mathcal{C}']texpr$	M	simul-sub all vars in $\mathcal{C}$ for all vars in $\mathcal{C}'$ in $texpr$
terminals	::=			
	|	λ		
	|	$\longrightarrow$		
	|	$\rightarrow$		
	|	$\rightsquigarrow$		
	|	$\Rightarrow$		
	|	$\Leftarrow$		
	|	$\vdash$		
	|	$\in$		
	|	Π		
	|	$\forall$		
	|	$\multimap$		
	|	$\cup$		
	|	Σ		
	|	$\exists$		
	|	$\star$		
	|	$\times$		
	|	$\wedge$		
	|	$\bigwedge$		
	|	$\neg$		
	|	$=$		

	$\neq$		
	$\leq$		
	$\geq$		
	$\&$		
	$.$		
	$\mid$		
	$+_{\text{ptr}}$		
	$\mapsto$		
	$*$		
	::		
	✓		
	:		
	.		
	.		
	$\gg$		
	::		
	$\hat{}$		
	$\bigvee$		
	$\equiv$		
	$\langle$		
	$\rangle$		
	$\Downarrow$		

z	::=			OCaml arbitrary-width integer
	|	i	M	literal integer
	|	to_int(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

	$\mid$	$\min\_int_\tau$	M minimum value of int of type τ
$\mathbb{Q}$, q, _	::=		OCaml type for rational numbers
	$\mid$	$\frac{int_1}{int_2}$	
lit	::=		
	$\mid$	*ident*	
	$\mid$	`unit`	
	$\mid$	*bool*	
	$\mid$	z	
	$\mid$	$\mathbb{Q}$	
arg	::=		argument/function types
	$\mid$	$\Pi\, ident{:}\beta.\ arg$	
	$\mid$	$\forall\, ident{:}\beta.\ arg$	
	$\mid$	$resource \multimap arg$	
	$\mid$	$term \supset arg$	
	$\mid$	*ret*	
	$\mid$	$[spine\_elem/ident]arg$	M
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
	$\mid$	$\Sigma\, ident{:}\beta.\ pure\_ret$	
	$\mid$	$term \wedge pure\_ret$	
	$\mid$	`I`	

$\mathcal{C}$	::=			computational var env
	|	.		
	|	$\mathcal{C}, ident{:}\beta$		
	|	$\overline{\mathcal{C}_i}^{\,i}$		
	|	$\mathrm{fresh}(\mathcal{C})$	M	identical context except with fresh variable names
$\mathcal{L}$	::=			logical var env
	|	.		
	|	$\overline{\mathcal{L}_i}^{\,i}$		
	|	$\mathcal{L}, ident{:}\beta$		
	|	$[\mathcal{C}/\mathcal{C}']\mathcal{L}$	M	
Φ	::=			constraints env
	|	.		
	|	$\Phi, term$		
	|	$\overline{\Phi_i}^{\,i}$		
	|	$[\mathcal{C}/\mathcal{C}']\Phi$	M	
$\mathcal{R}$	::=			resources env
	|	.		
	|	$\mathcal{R}, ident{:}resource$		
	|	$\overline{\mathcal{R}_i}^{\,i}$		
	|	$[\mathcal{C}/\mathcal{C}']\mathcal{R}$	M	
formula	::=			
	|	*judgement*		
	|	$abbrev \equiv term$		
	|	$\texttt{smt}\,(\Phi \Rightarrow term)$		
	|	$ident{:}\beta \in \mathcal{C}$		
	|	$ident{:}\,\texttt{struct}\,tag\ \&\ \overline{member_i{:}\tau_i}^{\,i} \in \texttt{Globals}$		

	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash mem\_val_i \Rightarrow \texttt{mem}\,\beta_i}^{\,i}$	dependent on memory object model
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash pval_i \Rightarrow \beta_i}^{\,i}$	
	$name{:}arg \in \texttt{Globals}$	
	$\overline{term_i\,\texttt{as}\,pattern_i{:}\beta_i \rightsquigarrow \mathcal{C}_i; \Phi_i}^{\,i}$	
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash tpexpr_i \Leftarrow y_i{:}\beta_i.\ term_i}^{\,i}$	
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i; \mathcal{R}_i \vdash texpr_i \Leftarrow ret_i}^{\,i}$	
	$\mathcal{L} \vdash term{:}\beta$	
	$pval{:}arg \in \texttt{Globals}$	

object_value_jtype ::=

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

pval_jtype ::=

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

resource_jtype ::=

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

spine_jtype ::=

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

pexpr_jtype ::=

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

pattern_jtype ::=

	$term\,\texttt{as}\,pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi$
	$term\,\texttt{as}\,ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi$
	$\overline{ret\_pattern_i}^{\,i} {:} ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}$
	$res\_pattern{:}resource \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}$

$$
\begin{array}{lll}
\mathit{tpval\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpexpr\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{action\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{tval\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{texpr\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret} \\
 & & \\
\mathit{opsem\_jtype} & ::= & \\
 & | & \langle \mathit{pexpr} \rangle \Downarrow \langle \mathit{pval} \rangle \\
 & | & \langle \mathit{tpexpr} \rangle \Downarrow \langle \mathit{tpval} \rangle \\
 & & \\
\mathit{judgement} & ::= & \\
 & | & \mathit{object\_value\_jtype} \\
 & | & \mathit{pval\_jtype} \\
 & | & \mathit{resource\_jtype} \\
 & | & \mathit{spine\_jtype} \\
 & | & \mathit{pexpr\_jtype} \\
 & | & \mathit{pattern\_jtype}
\end{array}
$$

| *tpval_jtype*
| *tpexpr_jtype*
| *action_jtype*
| *tval_jtype*
| *texpr_jtype*
| *opsem_jtype*

user_syntax ::=
| *ident*
| *n*
| *impl_const*
| *mem_int*
| *member*
|
| *nat*
| *mem_ptr*
| *mem_val*
|
| *mem_iv_c*
| *UB_name*
| *string*
|
|
| *mem_order*
| *linux_mem_order*
|
| *Sctypes_t*
| *tag*
| β
| *binop*

| $binop_{arith}$
| $binop_{rel}$
| $binop_{bool}$
| $ident$
| τ
| $ident$
| $object\_value$
| $loaded\_value$
| β
| $value$
| $bool\_value$
| $ctor\_val$
| $ctor\_expr$
| τ
| $name$
| $pval$
| $pval$
| $pexpr$
| $pexpr$
| $tpval$
| $tpval$
| $ident\_opt\_\beta$
| $pattern$
| $pattern$
| $ident\_or\_pattern$
| $tpexpr$
| $tpexpr\_case\_branch$
| $tpexpr$
| m_kill_kind

| *bool*
| *int*
| *mem_action*
| *mem_action*
| *polarity*
| *pol_mem_action*
| *mem_op*
| *res_term*
| *spine_elem*
| *tval*
| *tval*
| *res_pattern*
| *ret_pattern*
| *bool_op*
| *arith_op*
| *cmp_op*
| *list_op*
| *tuple_op*
| *pointer_op*
| β
| *option_op*
| *array_op*
| *param_op*
| *struct_op*
| *ct_pred*
| *term*
| *term*
| *init*
| *points_to*

| *resource*
| *ret*
| *rets*
| *seq_expr*
| *seq_expr*
| *seq_texpr*
| *texpr_case_branch*
| *seq_texpr*
| *is_expr*
| *is_expr*
| *is_texpr*
| *is_texpr*
| *texpr*
| *terminals*
| z
| $\mathbb{Q}$
| *lit*
| *arg*
| *pure_arg*
| *pure_ret*
| $\mathcal{C}$
| $\mathcal{L}$
| Φ
| $\mathcal{R}$
| *formula*

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

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

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

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{loaded\_value}_i \Rightarrow \beta}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{array}\,(\,\overline{\mathit{loaded\_value}_i}^{\,i}\,) \Rightarrow \texttt{obj}\,\texttt{array}\,\beta}\quad \textsc{Pval\_Obj\_Arr}$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow \texttt{list}\,\beta\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Cons}(pval_1, pval_2) \Rightarrow \texttt{list}\,\beta} \quad \textsc{Pval\_Ctor\_Cons}$$

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\Phi \vdash resource_1 \equiv resource_1' \\ \Phi \vdash resource_2 \equiv resource_2'\end{array}}{\Phi \vdash resource_1 \star resource_2 \equiv resource_1' \star resource_2'} \quad \textsc{Resource\_Eq\_SepConj}$$

$$\frac{\Phi \vdash resource \equiv resource'}{\Phi \vdash \exists\, ident{:}\beta.\ resource \equiv \exists\, ident{:}\beta.\ resource'} \quad \textsc{Resource\_Eq\_Exists}$$

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1 \vdash res\_term_1 \Leftarrow resource_1 \\ \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_2 \vdash res\_term_2 \Leftarrow resource_2\end{array}}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}_1, \mathcal{R}_2 \vdash \langle res\_term_1, res\_term_2 \rangle \Leftarrow resource_1 \star resource_2} \quad \textsc{Resource\_SepConj}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{res\_term}_2 \Leftarrow [\mathit{pval}/y]\mathit{resource}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathtt{pack}\,(\mathit{pval}, \mathit{res\_term}_2) \Leftarrow \exists\, y{:}\beta.\ \mathit{resource}}\ \textsc{Resource\_Pack}$$

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: [\mathit{pval}/x]\mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{pval},\ \overline{\mathit{spine\_elem}_i}^{\,i} :: \Pi\, x{:}\beta.\ \mathit{arg} \gg \mathit{ret}}\ \textsc{Spine\_Computational}$$

$$\frac{\begin{array}{l}\mathcal{L} \vdash \mathit{term}{:}\beta \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: [\mathit{term}/x]\mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{term},\ \overline{\mathit{spine\_elem}_i}^{\,i} :: \forall\, x{:}\beta.\ \mathit{arg} \gg \mathit{ret}}\ \textsc{Spine\_Logical}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash \mathit{res\_term} \Leftarrow \mathit{resource} \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_2 \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1,\mathcal{R}_2 \vdash \mathit{res\_term},\ \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{resource} \multimap \mathit{arg} \gg \mathit{ret}}\ \textsc{Spine\_Resource}$$

$$\frac{\begin{array}{l}\mathtt{smt}\,(\Phi \Rightarrow \mathit{term}) \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{term} \supset \mathit{arg} \gg \mathit{ret}}\ \textsc{Spine\_Constraint}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{}{term\,\texttt{as}\,\_{:}\beta{:}\beta \rightsquigarrow \cdot;\cdot}\ \textsc{Comp\_Pattern\_No\_Sym\_Annot}$$

$$\frac{}{term\,\texttt{as}\,x{:}\beta{:}\beta \rightsquigarrow \cdot, x{:}\beta;\cdot, x = term}\ \textsc{Comp\_Pattern\_Sym\_Annot}$$

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

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

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

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

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

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

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

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

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

$$\frac{}{:\texttt{I} \rightsquigarrow \cdot; \cdot; \cdot; \cdot}\ \textsc{Ret\_Pattern\_Empty}$$

$$\frac{\begin{array}{c} y\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \overline{ret\_pattern_i}^{\,i} : ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\texttt{comp}\ ident\_or\_pattern,\ \overline{ret\_pattern_i}^{\,i} : \Sigma\ y{:}\beta.\ ret \rightsquigarrow \mathcal{C}_1, \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2}\ \textsc{Ret\_Pattern\_Computational}$$

$$\frac{\overline{ret\_pattern_i}^{\,i} : ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\texttt{log}\ y,\ \overline{ret\_pattern_i}^{\,i} : \exists\ y{:}\beta.\ ret \rightsquigarrow \mathcal{C}; \mathcal{L}, y{:}\beta; \Phi; \mathcal{R}}\ \textsc{Ret\_Pattern\_Logical}$$

$$\frac{\begin{array}{c} res\_pattern{:}resource \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ \overline{ret\_pattern_i}^{\,i} : ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\texttt{res}\ res\_pattern,\ \overline{ret\_pattern_i}^{\,i} : resource \star ret \rightsquigarrow \mathcal{C}_2; \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2}\ \textsc{Ret\_Pattern\_Resource}$$

$$\frac{\overline{ret\_pattern_i}^{\,i} : ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\overline{ret\_pattern_i}^{\,i} : term \wedge ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi, term; \mathcal{R}} \quad \textsc{Ret\_Pattern\_Constraint}$$

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

$$\frac{}{\texttt{emp}:\texttt{emp} \rightsquigarrow \cdot; \cdot; \cdot} \quad \textsc{Res\_Pattern\_Empty}$$

$$\frac{}{\texttt{pt}:points\_to \rightsquigarrow \cdot; \cdot; \cdot, r:points\_to} \quad \textsc{Res\_Pattern\_PointsTo}$$

$$\frac{}{r:resource \rightsquigarrow \cdot; \cdot; \cdot, r:resource} \quad \textsc{Res\_Pattern\_Var}$$

$$\frac{\begin{array}{c} res\_pattern_1 : resource_1 \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ res\_pattern_2 : resource_2 \rightsquigarrow \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{\langle res\_pattern_1, res\_pattern_2 \rangle : resource_1 \star resource_2 \rightsquigarrow \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2} \quad \textsc{Res\_Pattern\_SepConj}$$

$$\frac{res\_pattern : resource \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R}}{res\_pattern : term \wedge resource \rightsquigarrow \mathcal{L}; \Phi, term; \mathcal{R}} \quad \textsc{Res\_Pattern\_Conj}$$

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

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

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta_1 \\ \dfrac{}{y_1\,\texttt{as}\,pattern_i{:}\beta_1 \rightsquigarrow \mathcal{C}_i;\Phi_i}\,i \\ \dfrac{}{\mathcal{C}, \mathrm{fresh}(\mathcal{C}_i);\mathcal{L}, y_1{:}\beta_1;\Phi, y_1 = pval, [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]\Phi_i \vdash [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]tpexpr_i \Leftarrow y_2{:}\beta_2.\ term_2}\,i\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{case}\,pval\,\texttt{of}\,\overline{\mid\ pattern_i \Rightarrow tpexpr_i}^{\,i}\,\texttt{end} \Leftarrow y_2{:}\beta_2.\ term_2}\ \textsc{TPExpr\_Case}$$

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l}\texttt{smt}\,(\Phi \Rightarrow term) \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow term \wedge ret}\ \textsc{TVal\_Constraint}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1 \vdash res\_term \Leftarrow resource \\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}_2 \vdash \texttt{done}\ \overline{spine\_elem_i}^{\,i} \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1, \mathcal{R}_2 \vdash \texttt{done}\, res\_term,\ \overline{spine\_elem_i}^{\,i} \Leftarrow resource \star ret}\ \textsc{TVal\_Resource}$$

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pexpr} \Rightarrow y{:}\beta.\ \mathit{term}}{\mathcal{C};\mathcal{L};\Phi;\cdot \vdash \mathit{pval} \Rightarrow \Sigma\, y{:}\beta.\ \mathit{term} \wedge \texttt{I}}\ \textsc{Seq\_Expr\_Pure}$$

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

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

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

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

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

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

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{Pos}\, mem\_action \Rightarrow ret}\ \textsc{IS\_Expr\_Action}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash mem\_action \Rightarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{Neg}\, mem\_action \Rightarrow ret}\ \textsc{IS\_Expr\_Neg\_Action}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_i \vdash texpr_i \Leftarrow \Sigma\, y_i{:}\beta_i.\ ret_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi;\overline{\mathcal{R}_i}^{\,i} \vdash \texttt{unseq}\,(\overline{texpr_i{:}\Sigma\, y_i{:}\beta_i.\ ret_i}^{\,i}) \Rightarrow \Sigma\, y{:}\overline{\beta_i}^{\,i}.\ \overline{[y^{(i)}/y_i]ret_i}^{\,i}}\ \textsc{IS\_Expr\_Unseq}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash tval \Leftarrow ret}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash tval \Leftarrow ret}\ \textsc{Seq\_TExpr\_TVal}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash seq\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C}, \text{fresh}(\mathcal{C}_1);\mathcal{L},\mathcal{L}_1;\Phi,[\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1;\mathcal{R},[\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathcal{R}_1 \vdash [\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]texpr \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr \Leftarrow ret_2}\ \textsc{Seq\_TExpr\_Let}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash texpr_1 \Leftarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C}, \text{fresh}(\mathcal{C}_1);\mathcal{L},\mathcal{L}_1;\Phi,[\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1;\mathcal{R},[\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathcal{R}_1 \vdash [\text{fresh}(\mathcal{C}_1)/\mathcal{C}_1]texpr_2 \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}', \mathcal{R} \vdash \texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret_1 = texpr_1\, \texttt{in}\, texpr_2 \Leftarrow ret_2}\ \textsc{Seq\_TExpr\_LetT}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \beta_1 \\ \overline{y_1\,\texttt{as}\,pattern_i{:}\beta_1 \rightsquigarrow \mathcal{C}_i;\Phi_i}^{\,i} \\ \overline{\mathcal{C},\mathrm{fresh}(\mathcal{C}_i);\mathcal{L},y_1{:}\beta_1;\Phi,y_1 = pval,[\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]\Phi_i;\mathcal{R} \vdash [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]texpr_i \Leftarrow ret}^{\,i}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{case}\,pval\,\texttt{of}\,\overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\,\texttt{end} \Leftarrow ret}\ \textsc{Seq\_TExpr\_Case}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval \Rightarrow \texttt{bool} \\ \mathcal{C};\mathcal{L};\Phi,pval = \texttt{true};\mathcal{R}_1 \vdash texpr_1 \Leftarrow ret \\ \mathcal{C};\mathcal{L};\Phi,pval = \texttt{false};\mathcal{R}_2 \vdash texpr_2 \Leftarrow ret\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}_1,\mathcal{R}_2 \vdash \texttt{if}\,pval\,\texttt{then}\,texpr_1\,\texttt{else}\,texpr_2 \Leftarrow ret}\ \textsc{Seq\_TExpr\_If}$$

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathrm{fresh}(\mathcal{C}_1);\mathcal{L},\mathcal{L}_1;\Phi,[\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1;\mathcal{R},[\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathcal{R}_1 \vdash [\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]texpr \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{let}\,\texttt{weak}\,\overline{ret\_pattern_i}^{\,i} = is\_expr\,\texttt{in}\,texpr \Leftarrow ret_2}\ \textsc{IS\_TExpr\_LetWeak}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash is\_expr \Rightarrow ret_1 \\ \overline{ret\_pattern_i}^{\,i} : ret_1 \rightsquigarrow \mathcal{C}_1;\mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C},\mathrm{fresh}(\mathcal{C}_1);\mathcal{L},\mathcal{L}_1;\Phi,[\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1;\mathcal{R},[\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathcal{R}_1 \vdash [\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]texpr \Leftarrow ret_2\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{let}\,\texttt{strong}\,\overline{ret\_pattern_i}^{\,i} = is\_expr\,\texttt{in}\,texpr \Leftarrow ret_2}\ \textsc{IS\_TExpr\_LetStrong}$$

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

$$\frac{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_texpr \Leftarrow ret}{\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_texpr \Leftarrow ret} \quad \textsc{TExpr\_IS}$$

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

$\boxed{\langle pexpr \rangle \Downarrow \langle pval \rangle}$

$$\frac{}{\langle \texttt{array\_shift}\,(mem\_ptr, \tau, mem\_int) \rangle \Downarrow \langle mem\_ptr' \rangle} \quad \textsc{PExpr\_Big\_ArrayShift}$$

$$\frac{}{\langle \texttt{member\_shift}\,(mem\_ptr, tag, member) \rangle \Downarrow \langle mem\_ptr' \rangle} \quad \textsc{PExpr\_Big\_MemberShift}$$

$$\frac{}{\langle \texttt{not}\,(\texttt{True}) \rangle \Downarrow \langle \texttt{False} \rangle} \quad \textsc{PExpr\_Big\_NotTrue}$$

$$\frac{}{\langle \texttt{not}\,(\texttt{False}) \rangle \Downarrow \langle \texttt{True} \rangle} \quad \textsc{PExpr\_Big\_NotFalse}$$

$$\frac{}{\langle mem\_int_1\ binop_{arith}\ mem\_int_2 \rangle \Downarrow \langle mem\_int \rangle} \quad \textsc{PExpr\_Big\_Arith\_Binop}$$

$$\frac{}{\langle mem\_int_1\ binop_{rel}\ mem\_int_2 \rangle \Downarrow \langle bool\_value \rangle} \quad \textsc{PExpr\_Big\_Rel\_Binop}$$

$$\frac{}{\langle bool\_value_1\ binop_{bool}\ bool\_value_2 \rangle \Downarrow \langle bool\_value \rangle} \quad \textsc{PExpr\_Big\_Bool\_Binop}$$

$$\frac{\langle tpexpr \rangle \Downarrow \langle \texttt{done}\ pval \rangle}{\langle name(\overline{pval_i}^{\,i}) \rangle \Downarrow \langle pval \rangle}\ \textsc{PExpr\_Big\_Call}$$

$$\frac{}{\langle \texttt{assert\_undef}\,(\texttt{True},\ UB\_name) \rangle \Downarrow \langle \texttt{Unit} \rangle}\ \textsc{PExpr\_Big\_Assert\_Undef}$$

$$\frac{}{\langle \texttt{bool\_to\_integer}\,(\texttt{True}) \rangle \Downarrow \langle mem\_int \rangle}\ \textsc{PExpr\_Big\_Bool\_To\_Integer\_True}$$

$$\frac{}{\langle \texttt{bool\_to\_integer}\,(\texttt{False}) \rangle \Downarrow \langle mem\_int \rangle}\ \textsc{PExpr\_Big\_Bool\_To\_Integer\_False}$$

$$\frac{}{\langle \texttt{wrapI}\,(\tau, mem\_int) \rangle \Downarrow \langle mem\_int' \rangle}\ \textsc{PExpr\_Big\_WrapI}$$

$\boxed{\langle tpexpr \rangle \Downarrow \langle tpval \rangle}$

$$\frac{\langle tpexpr_1 \rangle \Downarrow \langle tpval_1 \rangle}{\langle \texttt{if True then}\ tpexpr_1\ \texttt{else}\ tpexpr_2 \rangle \Downarrow \langle tpval_1 \rangle}\ \textsc{TPExpr\_Big\_If\_True}$$

$$\frac{\langle tpexpr_2 \rangle \Downarrow \langle tpval_2 \rangle}{\langle \texttt{if False then}\ tpexpr_1\ \texttt{else}\ tpexpr_2 \rangle \Downarrow \langle tpval_2 \rangle}\ \textsc{TPExpr\_Big\_If\_False}$$

```
Definition rules:        115 good    0 bad
Definition rule clauses: 257 good    0 bad
```