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
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

	`IvOR`	bitwise OR
	`IvXOR`	bitwise XOR
	`Fvfromint`	cast integer to floating value
	`Ivfromfloat`	cast floating to integer value

name	::=	
	ident	Core identifier
	impl_const	implementation-defined constant

pval	::=	pure values
	ident	Core identifier
	impl_const	implementation-defined constant
	value	Core values
	`constrained` $(\overline{mem\_iv\_c_i, pval_i}^{\,i})$	constrained value
	`error` $(string, pval)$	impl-defined static error
	$ctor\_val(\overline{pval_i}^{\,i})$	data constructor application
	$(\texttt{struct}\, ident)\{\overline{.member_i = pval_i}^{\,i}\}$	C struct expression
	$(\texttt{union}\, ident)\{.member = pval\}$	C union expression

pexpr	::=	pure expressions
	pval	pure values
	$ctor\_expr(\overline{pval_i}^{\,i})$	data constructor application
	`array_shift` $(pval_1, \tau, pval_2)$	pointer array shift
	`member_shift` $(pval, ident, member)$	pointer struct/union member shift
	`not` $(pval)$	boolean not
	$pval_1\ binop\ pval_2$	binary operations
	`memberof` $(ident, member, pval)$	C struct/union member access
	$name(\overline{pval_i}^{\,i})$	pure function call
	`assert_undef` $(pval,\ UB\_name)$	
	`bool_to_integer` $(pval)$	

	|	$\texttt{conv\_int}\,(\tau, pval)$		
	|	$\texttt{wrapI}\,(\tau, pval)$		
tpval	::=			top-level pure values
	|	$\texttt{undef}\ UB\_name$		undefined behaviour
	|	$\texttt{done}\,pval$		pure done
ident_opt_β	::=			type annotated optional identifier
	|	$\_{:}\beta$		
	|	$ident{:}\beta$		
pattern	::=			
	|	$ident\_opt\_\beta$		
	|	$ctor\_val(\overline{pattern_i}^{\,i})$		
ident_or_pattern	::=			
	|	$ident$		
	|	$pattern$		
tpexpr	::=			top-level pure expressions
	|	$tpval$		top-level pure values
	|	$\texttt{case}\,pval\,\texttt{of}\,\overline{\mid pattern_i \Rightarrow tpexpr_i}^{\,i}\,\texttt{end}$		pattern matching
	|	$\texttt{let}\,ident\_or\_pattern = pexpr\,\texttt{in}\,tpexpr$		pure let
	|	$\texttt{if}\,pval\,\texttt{then}\,tpexpr_1\,\texttt{else}\,tpexpr_2$		pure if
	|	$[\mathcal{C}/\mathcal{C}']tpexpr$	M	simul-sub all vars in $\mathcal{C}$ for all vars in $\mathcal{C}'$ in $tpexpr$
m_kill_kind	::=			
	|	$\texttt{dynamic}$		
	|	$\texttt{static}\,\tau$		

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
	|	$pval_1 \not\equiv pval_2$	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
	|	`pt`	single-cell heap
	|	*ident*	variable
	|	$\langle res\_term_1, res\_term_2 \rangle$	seperating-conjunction pair
	|	`pack` $(pval, res\_term_2)$	packing for existentials
spine_elem	::=		spine element
	|	*pval*	pure value

	$term$	logical value
	res_term	resource value

$tval$	::=		(effectful) top-level values
	|	$\texttt{done}\ \overline{spine\_elem_i}^{\,i}$	end of top-level expression
	|	$\texttt{undef}\ UB\_name$	undefined behaviour

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

$ret\_pattern$	::=		return pattern
	|	$\texttt{comp}\ ident\_or\_pattern$	computational variable
	|	$\texttt{log}\ ident$	logical variable
	|	$\texttt{res}\ res\_pattern$	resource variable

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

$arith\_op$	::=		
	|	$term_1 + term_2$	
	|	$term_1 - term_2$	

	$\mid$	$term_1 \times term_2$	
	$\mid$	$term_1 / term_2$	
	$\mid$	$term_1$ `rem_t` $term_2$	
	$\mid$	$term_1$ `rem_f` $term_2$	
	$\mid$	$term_1$ `^` $term_2$	
	$\mid$	$term_1\ binop_{arith}\ term_2$	M
cmp_op	::=		
	$\mid$	$term_1 < term_2$	less than
	$\mid$	$term_1 \leq term_2$	less than or equal
	$\mid$	$term_1\ binop_{rel}\ term_2$	M
list_op	::=		
	$\mid$	`nil`	
	$\mid$	`tl` $term$	
	$\mid$	$term^{(int)}$	
tuple_op	::=		
	$\mid$	$(\overline{term_i}^{\,i})$	
	$\mid$	$term^{(int)}$	
pointer_op	::=		
	$\mid$	mem_ptr	
	$\mid$	$term_1 +_{\mathrm{ptr}} term_2$	
	$\mid$	`cast_int_to_ptr` $term$	
	$\mid$	`cast_ptr_to_int` $term$	
option_op	::=		
	$\mid$	`none` β	
	$\mid$	`some` $term$	

array_op	::=			
	|	$term_1[term_2]$		
param_op	::=			
	|	$ident{:}\beta.\, term$		
	|	$term(term_1, .., term_n)$		
struct_op	::=			
	|	$term.member$		
ct_pred	::=			
	|	$\texttt{representable}\,(\tau, term)$		
	|	$\texttt{aligned}\,(\tau, term)$		
	|	$\texttt{alignedI}\,(term_1, term_2)$		
term, _	::=			
	|	*lit*		
	|	*arith_op*		
	|	*bool_op*		
	|	*cmp_op*		
	|	*tuple_op*		
	|	*struct_op*		
	|	*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
	$\Sigma \, ident{:}\beta.\ ret$		
	$\exists \, ident{:}\beta.\ ret$		
	$resource \star ret$		
	$term \wedge ret$		
	`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
	|	$\texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr$	bind return patterns
	|	$\texttt{let}\, \overline{ret\_pattern_i}^{\,i} : ret = texpr_1\, \texttt{in}\, texpr_2$	annotated bind return patterns
	|	$\texttt{case}\, pval\, \texttt{of}\, \overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\, \texttt{end}$	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 runtime
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
	|	$\texttt{let}\, \texttt{weak}\, pattern = is\_expr\, \texttt{in}\, texpr$	weak sequencing
	|	$\texttt{let}\, \texttt{strong}\, ident\_or\_pattern = is\_expr\, \texttt{in}\, texpr$	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	

$$
\begin{array}{ll}
| & \longrightarrow \\
| & \rightarrow \\
| & \leadsto \\
| & \Rightarrow \\
| & \Leftarrow \\
| & \vdash \\
| & \in \\
| & \Pi \\
| & \forall \\
| & \multimap \\
| & \supset \\
| & \Sigma \\
| & \exists \\
| & \star \\
| & \times \\
| & \wedge \\
| & \bigwedge \\
| & \neg \\
| & = \\
| & \neq \\
| & \leq \\
| & \geq \\
| & \& \\
| & . \\
| & | \\
| & +_{\text{ptr}} \\
| & \mapsto \\
| & * \\
| & ::
\end{array}
$$

	✓		
	:		
	.		
	.		
	≫		
	::		
	^		
	⋁		
	≡		
	⟨		
	⟩		

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
	|	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
	|	$\mathbb{Q}$

arg	::=			argument/function types
	|	$\Pi\, ident{:}\beta.\ arg$		
	|	$\forall\, ident{:}\beta.\ arg$		
	|	$resource \multimap arg$		
	|	$term \supset arg$		
	|	ret		
	|	$[spine\_elem/ident]arg$	M	
$pure\_arg$	::=			pure argument/function types
	|	$\Pi\, ident{:}\beta.\ pure\_arg$		
	|	$term \supset pure\_arg$		
	|	$pure\_ret$		
$pure\_ret$	::=			pure return types
	|	$\Sigma\, ident{:}\beta.\ pure\_ret$		
	|	$term \wedge pure\_ret$		
	|	$\mathtt{I}$		
$\mathcal{C}$	::=			computational var env
	|	$.$		
	|	$\mathcal{C}, ident{:}\beta$		
	|	$\overline{\mathcal{C}_i}^{\,i}$		
	|	$\mathrm{fresh}(\mathcal{C})$	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	::=		
	|	$\mathcal{C}; \mathcal{L}; \Phi \vdash object\_value \Rightarrow \texttt{obj}\,\beta$	

$$
\begin{array}{lll}
\mathit{pval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \beta \\
 & & \\
\mathit{resource\_jtype} & ::= & \\
 & \mid & \Phi \vdash \mathit{resource} \equiv \mathit{resource}' \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{res\_term} \Leftarrow \mathit{resource} \\
 & & \\
\mathit{spine\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{\,i} :: \mathit{arg} \gg \mathit{ret} \\
 & & \\
\mathit{pexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{pattern\_jtype} & ::= & \\
 & \mid & \mathit{term}\ \mathtt{as}\ \mathit{pattern}{:}\beta \rightsquigarrow \mathcal{C}; \Phi \\
 & \mid & \mathit{term}\ \mathtt{as}\ \mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}; \Phi \\
 & \mid & \overline{\mathit{ret\_pattern}_i}^{\,i} {:}\mathit{ret} \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \\
 & \mid & \mathit{res\_pattern}{:}\mathit{resource} \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R} \\
 & & \\
\mathit{tpval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{tpexpr\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term} \\
 & & \\
\mathit{action\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{tval\_jtype} & ::= & \\
 & \mid & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{tval} \Leftarrow \mathit{ret}
\end{array}
$$

$$
\begin{array}{lll}
\mathit{texpr\_jtype} & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{seq\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{is\_texpr} \Leftarrow \mathit{ret} \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{texpr} \Leftarrow \mathit{ret}
\end{array}
$$

$$
\begin{array}{lll}
\mathit{judgement} & ::= & \\
 & | & \mathit{object\_value\_jtype} \\
 & | & \mathit{pval\_jtype} \\
 & | & \mathit{resource\_jtype} \\
 & | & \mathit{spine\_jtype} \\
 & | & \mathit{pexpr\_jtype} \\
 & | & \mathit{pattern\_jtype} \\
 & | & \mathit{tpval\_jtype} \\
 & | & \mathit{tpexpr\_jtype} \\
 & | & \mathit{action\_jtype} \\
 & | & \mathit{tval\_jtype} \\
 & | & \mathit{texpr\_jtype}
\end{array}
$$

$$
\begin{array}{lll}
\mathit{user\_syntax} & ::= & \\
 & | & \mathit{ident} \\
 & | & n \\
 & | & \mathit{impl\_const} \\
 & | & \mathit{mem\_int} \\
 & | & \mathit{member} \\
 & | & \\
 & | & \mathit{nat} \\
 & | & \mathit{mem\_ptr} \\
 & | & \mathit{mem\_val}
\end{array}
$$

| |
| | 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
| | $value$
| | $ctor\_val$
| | $ctor\_expr$
| | τ
| | $name$
| | $pval$
| | $pval$

| *pexpr*
| *pexpr*
| *tpval*
| *tpval*
| *ident_opt_β*
| *pattern*
| *pattern*
| *ident_or_pattern*
| *tpexpr*
| *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
| 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*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \mathtt{loc} \quad \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \mathtt{integer}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathtt{array\_shift}\,(\mathit{pval}_1,\tau,\mathit{pval}_2) \Rightarrow y{:}\mathtt{loc}.\ y = \mathit{pval}_1 +_{\mathrm{ptr}} (\mathit{pval}_2 \times \mathrm{size\_of}(\tau))}\ \textsc{PExpr\_Array\_Shift}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \mathtt{loc} \quad \_{:}\,\mathtt{struct}\,\mathit{tag}\ \&\,\overline{\mathit{member}_i{:}\tau_i}^{\,i} \in \mathtt{Globals}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathtt{member\_shift}\,(\mathit{pval},\mathit{tag},\mathit{member}_j) \Rightarrow y{:}\mathtt{loc}.\ y = \mathit{pval} +_{\mathrm{ptr}} \mathrm{offset\_of}_{\mathit{tag}}(\mathit{member}_j)}\ \textsc{PExpr\_Member\_Shift}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \mathtt{integer} \quad \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \mathtt{integer}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1\ \mathit{binop}_{\mathit{arith}}\ \mathit{pval}_2 \Rightarrow y{:}\mathtt{integer}.\ y = (\mathit{pval}_1\ \mathit{binop}_{\mathit{arith}}\ \mathit{pval}_2)}\ \textsc{PExpr\_Arith\_Binop}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\overline{\mathit{ret\_pattern}_i}^{\,i} : \mathit{ret} \rightsquigarrow \mathcal{C};\mathcal{L};\Phi;\mathcal{R}}{\mathtt{log}\ y,\ \overline{\mathit{ret\_pattern}_i}^{\,i} : \exists\, y{:}\beta.\ \mathit{ret} \rightsquigarrow \mathcal{C};\mathcal{L},y{:}\beta;\Phi;\mathcal{R}} \quad \textsc{Ret\_Pattern\_Logical}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l} \mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow y_1{:}\beta_1.\ \mathit{term}_1 \\ y_1\ \texttt{as}\ \mathit{ident\_or\_pattern}{:}\beta_1 \rightsquigarrow \mathcal{C}_1; \Phi_1 \\ \mathcal{C}, \mathrm{fresh}(\mathcal{C}_1); \mathcal{L}, y_1{:}\beta_1; \Phi, \mathit{term}_1, [\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1 \vdash [\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\ \mathit{term}_2 \end{array}}{\mathcal{C}; \mathcal{L}; \Phi \vdash \texttt{let}\,\mathit{ident\_or\_pattern} = \mathit{pexpr}\,\texttt{in}\,\mathit{tpexpr} \Leftarrow y_2{:}\beta_2.\ \mathit{term}_2} \quad \textsc{TPExpr\_Let}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow \texttt{loc}\\ \texttt{smt}\,(\Phi \Rightarrow pval_0 = pval_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot, r{:}\langle pval_0 \xrightarrow{\checkmark}_{\tau} \_\rangle \vdash \texttt{memop}\,(\texttt{ptrValidForDeref}\,(\tau,pval_1)) \Rightarrow \Sigma\, y{:}\texttt{bool}.\ y = \texttt{aligned}\,(\tau,pval_0) \wedge \texttt{I}}\ \textsc{IS\_Expr\_Memop\_PtrValidForDeref}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Definition rules:        101 good    0 bad
Definition rule clauses: 240 good    0 bad
```