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
logical_val	logical values (to be specified)
	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
	|	$pval_1 \neq pval_2$	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` $(pval)$	
spine_elem	::=		spine element
	|	*pval*	pure value
	|	*logical_val*	logical variable
	|	*res_term*	resource valuel
tval	::=		(effectful) top-level values
	|	`done` $\overline{spine\_elem_i}^{\,i}$	end of top-level expression
	|	`undef` *UB_name*	undefined behaviour

bool_op ::=

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

arith_op ::=

|	$term_1 + term_2$	
|	$term_1 - term_2$	
|	$term_1 \times term_2$	
|	$term_1 / term_2$	
|	$term_1\ \texttt{rem\_t}\ term_2$	
|	$term_1\ \texttt{rem\_f}\ term_2$	
|	$term_1\ \texttt{\^{}}\ term_2$	
|	$term_1\ binop_{arith}\ term_2$	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 ::=

|	$(\overline{term_i}^{\,i})$	

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

	$struct\_op$		
	$pointer\_op$		
	$list\_op$		
	$array\_op$		
	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

$terms$	::=		non-empty list of terms
	$[term_1, \ldots, term_n]$		

$predicate\_name$	::=		names of predicates
	$Sctypes\_t$		C type
	$string$		arbitrary

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

$predicate$	::=		arbitrary predicate
	$terms_1 \, \mathbb{Q}^{init}_{\mapsto predicate\_name} \, terms_2$		

$resource$	::=		resources
	`emp`		empty heap
	$predicate$		heap predicate
	$term$		logical term
	$resource_1 \star resource_2$		seperating conjunction

	$\exists\, ident{:}\beta.\ resource$		existential
	$resource_1 \wedge resource_2$		logical conjuction
	$[pval/ident]resource$	M	substitute *pval* for *ident* in *resource*

res_term	::=		resource terms
	`emp`		empty heap
	ident		variable
	$\langle res\_term_1, res\_term_2 \rangle$		seperating-conjunction pair
	`pack` $(pval, res\_term_2)$		packing for existentials
	(res_term_1, res_term_2)		logical-conjunction pair

res_pattern	::=		resource terms
	`emp`		empty heap
	ident		variable
	$\langle res\_pattern_1, res\_pattern_2 \rangle$		seperating-conjunction pair
	`pack` $(ident, res\_pattern)$		packing for existentials
	$(res\_pattern_1, res\_pattern_2)$		logical-conjunction pair

ret_pattern	::=		return pattern
	`comp` *ident*		computational variable
	`log` *ident*		logical variable
	`res` *ident*		resource variable

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

seq_texpr	::=		sequential top-level (effectful) expressions
	tval		(effectful) top-level values

	$\mid$	$\texttt{run}\, ident\, pval_1, .., pval_n$	run from label
	$\mid$	$\texttt{nd}\, (pval_1, .., pval_n)$	nondeterministic choice
	$\mid$	$\texttt{let}\, \overline{ret\_pattern_i}^{\,i} = seq\_expr\, \texttt{in}\, texpr$	bind return patterns
	$\mid$	$\texttt{letC}\, ident\_or\_pattern = pval\, \texttt{in}\, texpr$	bind computational patterns
	$\mid$	$\texttt{letR}\, res\_pattern = res\_term\, \texttt{in}\, texpr$	bind resource patterns
	$\mid$	$\texttt{case}\, pval\, \texttt{of}\, \overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\, \texttt{end}$	pattern matching
	$\mid$	$\texttt{if}\, pval\, \texttt{then}\, texpr_1\, \texttt{else}\, texpr_2$	conditional
	$\mid$	$\texttt{bound}\, [int](is\_texpr)$	limit scope of indet seq behaviour, absent at runtime
is_expr	::=		indet seq (effectful) expressions
	$\mid$	$\texttt{memop}\, (mem\_op)$	pointer op involving memory
	$\mid$	pol_mem_action	memory action
	$\mid$	$\texttt{unseq}\, (texpr_1, .., texpr_n)$	unsequenced expressions
is_texpr	::=		indet seq top-level (effectful) expressions
	$\mid$	$\texttt{let weak}\, pattern = is\_expr\, \texttt{in mu\_texpr\_aux}$	weak sequencing
	$\mid$	$\texttt{let strong}\, ident\_or\_pattern = is\_expr\, \texttt{in mu\_texpr\_aux}$	strong sequencing
$texpr$	::=		top-level (effectful) expressions
	$\mid$	seq_texpr	sequential (effectful) expressions
	$\mid$	is_texpr	indet seq (effectful) expressions
	$\mid$	$[\mathcal{C}/\mathcal{C}']texpr$	M simul-sub all vars in $\mathcal{C}$ for all vars in $\mathcal{C}'$ in $texpr$
$terminals$	::=		
	$\mid$	λ	
	$\mid$	$\longrightarrow$	
	$\mid$	$\rightarrow$	
	$\mid$	$\rightsquigarrow$	
	$\mid$	$\Rightarrow$	
	$\mid$	$\Leftarrow$	

| $\vdash$

| $\in$

| Π

| $\forall$

| $\multimap$

| $\supset$

| Σ

| $\exists$

| $\star$

| $\times$

| $\wedge$

| $\bigwedge$

| $\neg$

| $=$

| $\neq$

| $\leq$

| $\geq$

| &

| $\cdot$

| $|$

| $+_{\mathrm{ptr}}$

| $\mapsto$

| $*$

| $::$

| $\checkmark$

| $:$

| .

| .

| $\gg$

	::		
	^		
	$\bigvee$		
	$\equiv$		
	$\langle$		
	$\rangle$		

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

$\mathbb{Q}$	::=		OCaml type for rational numbers
	|	$\frac{int_1}{int_2}$	

lit	::=	
	|	*ident*
	|	`unit`
	|	*bool*
	|	z
	|	$\mathbb{Q}$

arg	::=		argument/function types
	|	$\Pi\, ident{:}\beta.\ arg$	
	|	$\forall\, ident{:}\beta.\ arg$	
	|	$resource \multimap arg$	

	$\mid$	$term \supset arg$	
	$\mid$	ret	
	$\mid$	$[spine\_elem/ident]arg$	M

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

$ret,\ \_$	::=			return types
	$\mid$	$\Sigma\, ident{:}\beta.\ ret$		
	$\mid$	$\exists\, ident{:}\beta.\ ret$		
	$\mid$	$resource \star ret$		
	$\mid$	$term \wedge ret$		
	$\mid$	$\mathtt{I}$		
	$\mid$	$[spine\_elem/ident]ret$	M	

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

$\mathcal{C}$	::=			computational var env
	$\mid$	.		
	$\mid$	$\mathcal{C}, ident{:}\beta$		
	$\mid$	$\overline{\mathcal{C}_i}^{\,i}$		
	$\mid$	$\mathrm{fresh}(\mathcal{C})$	M	identical context except with fresh variable names

$\mathcal{L}$	::=			logical var env
	$\mid$	.		

	$\overline{\mathcal{L}_i}^{\,i}$		
	$\mathcal{L}, ident{:}\beta$		
	$[\mathcal{C}/\mathcal{C}']\mathcal{L}$	M	
Φ	::=		constraints env
	$.$		
	$\Phi, term$		
	$\overline{\Phi_i}^{\,i}$		
	$[\mathcal{C}/\mathcal{C}']\Phi$	M	
$\mathcal{R}$	::=		resources env
	$.$		
	$\mathcal{R}, ident{:}resource$		
	$\mathcal{R}_1, \mathcal{R}_2$		
	$[\mathcal{C}/\mathcal{C}']\mathcal{R}$	M	
formula	::=		
	$judgement$		
	$abbrev \equiv term$		
	$\texttt{smt}\,(\Phi \Rightarrow term)$		
	$\texttt{smt}\,(\Phi \Rightarrow resource_1 = resource_2)$		
	$ident{:}\beta \in \mathcal{C}$		
	$ident{:}\,\texttt{struct}\,tag\;\&\;\overline{member_i{:}\tau_i}^{\,i} \in \texttt{Globals}$		
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash mem\_val_i \Rightarrow \texttt{mem}\,\beta_i}^{\,i}$		dependent on memory object model
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash pval_i \Rightarrow \beta_i}^{\,i}$		
	$name{:}arg \in \texttt{Globals}$		
	$\overline{term_i\,\texttt{as}\,pattern_i{:}\beta_i \rightsquigarrow \mathcal{C}_i; \Phi_i}^{\,i}$		
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash tpexpr_i \Leftarrow y_i{:}\beta_i.\; term_i}^{\,i}$		
	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i; \mathcal{R}_i \vdash texpr_i \Leftarrow ret_i}^{\,i}$		
	$\mathcal{L} \vdash logical\_val{:}\beta$		

$$
\begin{array}{lll}
 & | & pval{:}arg \in \texttt{Globals} \\
 & & \\
object\_value\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash object\_value \Rightarrow \texttt{obj}\ \beta \\
 & & \\
pval\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash pval \Rightarrow \beta \\
 & & \\
resource\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash res\_term \Rightarrow resource \\
 & & \\
spine\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{spine\_elem_i}^{\,i} :: arg \gg ret \\
 & & \\
pexpr\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash pexpr \Rightarrow ident{:}\beta.\, term \\
 & & \\
pattern\_jtype & ::= & \\
 & | & term\ \texttt{as}\ pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi \\
 & | & term\ \texttt{as}\ ident\_or\_pattern{:}\beta \rightsquigarrow \mathcal{C}; \Phi \\
 & | & \overline{ret\_pattern_i}^{\,i} {:}ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \\
 & | & res\_pattern{:}resource \rightsquigarrow \mathcal{L}; \Phi; \mathcal{R} \\
 & & \\
tpval\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash tpval \Leftarrow ident{:}\beta.\, term \\
 & & \\
tpexpr\_jtype & ::= & \\
 & | & \mathcal{C}; \mathcal{L}; \Phi \vdash tpexpr \Leftarrow ident{:}\beta.\, term \\
 & & \\
action\_jtype & ::= &
\end{array}
$$

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

$tval\_jtype$ ::=

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

$texpr\_jtype$ ::=

$$\mid \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq\_expr \Rightarrow ret$$
$$\mid \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_expr \Rightarrow ret$$
$$\mid \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash seq\_texpr \Leftarrow ret$$
$$\mid \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash is\_texpr \Leftarrow ret$$
$$\mid \quad \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash texpr \Leftarrow ret$$

$judgement$ ::=

$\mid \quad object\_value\_jtype$
$\mid \quad pval\_jtype$
$\mid \quad resource\_jtype$
$\mid \quad spine\_jtype$
$\mid \quad pexpr\_jtype$
$\mid \quad pattern\_jtype$
$\mid \quad tpval\_jtype$
$\mid \quad tpexpr\_jtype$
$\mid \quad action\_jtype$
$\mid \quad tval\_jtype$
$\mid \quad texpr\_jtype$

$user\_syntax$ ::=

$\mid \quad ident$
$\mid \quad n$
$\mid \quad impl\_const$
$\mid \quad mem\_int$

| $member$
|
| nat
| mem_ptr
| mem_val
|
| mem_iv_c
| UB_name
| $string$
|
|
| mem_order
| $linux\_mem\_order$
| $logical\_val$
|
| $Sctypes\_t$
| tag
| β
| $binop$
| $binop_{arith}$
| $binop_{rel}$
| $binop_{bool}$
| $ident$
| τ
| $ident$
| $object\_value$
| $loaded\_value$
| β
| $value$

| $ctor\_val$
| $ctor\_expr$
| τ
| $name$
| $pval$
| $pval$
| $pexpr$
| $pexpr$
| $tpval$
| $tpval$
| $ident\_opt\_\beta$
| $pattern$
| $pattern$
| $ident\_or\_pattern$
| $tpexpr$
| $tpexpr$
| m_kill_kind
| $bool$
| int
| mem_action
| mem_action
| $polarity$
| pol_mem_action
| mem_op
| $spine\_elem$
| $tval$
| $tval$
| $bool\_op$
| $arith\_op$

| cmp_op
| $list\_op$
| $tuple\_op$
| $pointer\_op$
| β
| $option\_op$
| $array\_op$
| $param\_op$
| $struct\_op$
| ct_pred
| $term$
| $term$
| $term$
| $terms$
| $predicate\_name$
| $init$
| $predicate$
| $resource$
| res_term
| $res\_pattern$
| $ret\_pattern$
| 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*
| *ret*
| *pure_ret*
| $\mathcal{C}$
| $\mathcal{L}$
| Φ
| $\mathcal{R}$
| *formula*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\texttt{smt}\,(\Phi \Rightarrow resource = resource')}{\mathcal{C};\mathcal{L};\Phi;\cdot, r{:}resource \vdash r \Rightarrow resource'}\ \textsc{Resource\_Var}$$

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

$$\frac{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{res\_term}_1 \Rightarrow \mathit{resource}_1 \quad \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{res\_term}_2 \Rightarrow \mathit{resource}_2}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash (\mathit{res\_term}_1, \mathit{res\_term}_2) \Rightarrow \mathit{resource}_1 \wedge \mathit{resource}_2}\ \textsc{Resource\_Conj}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \texttt{integer} \\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \texttt{integer}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1\ \mathit{binop}_{\mathit{rel}}\ \mathit{pval}_2 \Rightarrow y{:}\texttt{bool}.\ y = (\mathit{pval}_1\ \mathit{binop}_{\mathit{rel}}\ \mathit{pval}_2)}\ \textsc{PExpr\_Rel\_Binop}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \texttt{bool} \\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow \texttt{bool}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1\ \mathit{binop}_{\mathit{bool}}\ \mathit{pval}_2 \Rightarrow y{:}\texttt{bool}.\ y = (\mathit{pval}_1\ \mathit{binop}_{\mathit{bool}}\ \mathit{pval}_2)}\ \textsc{PExpr\_Bool\_Binop}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\overline{ret\_pattern_i}^{\,i} : ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}}{\texttt{res}\ y,\ \overline{ret\_pattern_i}^{\,i} : resource \star ret \rightsquigarrow \mathcal{C}; \mathcal{L}; \Phi; \mathcal{R}, y{:}resource} \quad \textsc{Ret\_Pattern\_Resource}$$

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

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

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

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

$$\frac{}{\_{:}term \rightsquigarrow \cdot; \cdot, term; \cdot} \quad \textsc{Res\_Pattern\_Var\_Term}$$

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

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

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

$$\frac{\begin{array}{c} res\_pattern_1{:}resource_1 \rightsquigarrow \mathcal{L}_1; \Phi_1; \mathcal{R}_1 \\ res\_pattern_2{:}resource_2 \rightsquigarrow \mathcal{L}_2; \Phi_2; \mathcal{R}_2 \end{array}}{(res\_pattern_1, res\_pattern_2){:}resource_1 \wedge resource_2 \rightsquigarrow \mathcal{L}_1, \mathcal{L}_2; \Phi_1, \Phi_2; \mathcal{R}_1, \mathcal{R}_2} \quad \textsc{Res\_Pattern\_Conj}$$

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

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \beta_1 \\ \hline y_1\ \texttt{as}\ \mathit{pattern}_i{:}\beta_1 \rightsquigarrow \mathcal{C}_i;\Phi_i\ {}^{i} \\ \hline \mathcal{C}, \mathrm{fresh}(\mathcal{C}_i);\mathcal{L}, y_1{:}\beta_1;\Phi, y_1 = \mathit{pval}, [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]\Phi_i \vdash [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]\mathit{tpexpr}_i \Leftarrow y_2{:}\beta_2.\ \mathit{term}_2\ {}^{i}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{case}\,\mathit{pval}\,\texttt{of}\ \overline{\mid\ \mathit{pattern}_i \Rightarrow \mathit{tpexpr}_i}^{\,i}\ \texttt{end} \Leftarrow y_2{:}\beta_2.\ \mathit{term}_2}\ \textsc{TPExpr\_Case}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \beta_1 \\ y_1\ \texttt{as}\ \mathit{ident\_or\_pattern}{:}\beta_1 \rightsquigarrow \mathcal{C}_1;\Phi_1 \\ \mathcal{C},\mathrm{fresh}(\mathcal{C}_1);\mathcal{L},y_1{:}\beta_1;\Phi,y_1 = \mathit{pval},[\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\Phi_1;\mathcal{R} \vdash [\mathrm{fresh}(\mathcal{C}_1)/\mathcal{C}_1]\mathit{texpr} \Leftarrow \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \texttt{letC}\ \mathit{ident\_or\_pattern} = \mathit{pval}\ \texttt{in}\ \mathit{texpr} \Leftarrow \mathit{ret}}\ \textsc{Seq\_TExpr\_Let\_Comp}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi;\mathcal{R}' \vdash \mathit{res\_term} \Rightarrow \mathit{resource}_1 \\ \mathit{res\_pattern}{:}\mathit{resource}_1 \rightsquigarrow \mathcal{L}_1;\Phi_1;\mathcal{R}_1 \\ \mathcal{C};\mathcal{L},\mathcal{L}_1;\Phi,\Phi_1;\mathcal{R},\mathcal{R}_1 \vdash \mathit{texpr} \Leftarrow \mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R}',\mathcal{R} \vdash \texttt{letR}\ \mathit{res\_pattern} = \mathit{res\_term}\ \texttt{in}\ \mathit{texpr} \Leftarrow \mathit{ret}}\ \textsc{Seq\_TExpr\_Let\_Res}$$

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

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

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

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

```
Definition rules:         86 good    0 bad
Definition rule clauses: 204 good    0 bad
```