ident, x, y, y_p, _	subscript p is for pointers
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 (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)
logical_val	logical values (to be specified)

$Sctypes\_t,\ \tau$	::=			C type
	|	$\tau *$		pointer to type τ
tag	::=			OCaml type for struct/union tag
	|	$ident$		
$\beta,\ \_$	::=			base types
	|	`unit`		unit
	|	`bool`		boolean
	|	`integer`		integer
	|	`real`		rational numbers?
	|	`loc`		location
	|	`array` β		array
	|	$[\beta]$		list
	|	$\overline{\beta_i}^{\,i}$		tuple
	|	`struct` tag		struct
	|	$\{\beta\}$		set
	|	`opt` (β)		option
	|	$\beta_1, \ldots, \beta_n \to \beta$		parameter types
	|	β_τ	M	of a C type
$binop$	::=			binary operators
	|	`+`		addition
	|	`-`		subtraction
	|	`*`		multiplication
	|	`/`		division
	|	`rem_t`		modulus
	|	`rem_f`		remainder
	|	`^`		exponentiation
	|	`=`		equality, defined both for integer and C types

	|	`>`	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

	|	$\texttt{array}\,(\,\overline{loaded\_value_i}^{\,i}\,)$	C array value
	|	$(\,\texttt{struct}\,ident)\{\,\overline{.\,member_i{:}\tau_i = mem\_val_i}^{\,i}\,\}$	C struct value
	|	$(\,\texttt{union}\,ident)\{\,.\,member = mem\_val\}$	C union value
$loaded\_value$	::=		potentially unspecified C object values
	|	$\texttt{specified}\,object\_value$	specified loaded value
$value$	::=		Core values
	|	$object\_value$	C object value
	|	$loaded\_value$	loaded C object value
	|	`Unit`	unit
	|	`True`	boolean true
	|	`False`	boolean false
	|	$\beta[\,\overline{value_i}^{\,i}\,]$	list
	|	$(\,\overline{value_i}^{\,i}\,)$	tuple
$ctor\_val$	::=		data constructors
	|	$\texttt{Nil}\,\beta$	empty list
	|	`Cons`	list cons
	|	`Tuple`	tuple
	|	`Array`	C array
	|	`Specified`	non-unspecified loaded value
$ctor\_expr$	::=		data constructors
	|	`Ivmax`	max integer value
	|	`Ivmin`	min integer value
	|	`Ivsizeof`	sizeof value
	|	`Ivalignof`	alignof value
	|	`IvCOMPL`	bitwise complement
	|	`IvAND`	bitwise AND

	|	`IvOR`	bitwise OR
	|	`IvXOR`	bitwise XOR
	|	`Fvfromint`	cast integer to floating value
	|	`Ivfromfloat`	cast floating to integer value
name	::=		
	|	*ident*	Core identifier
	|	*impl_const*	implementation-defined constant
pval	::=		pure values
	|	*ident*	Core identifier
	|	*impl_const*	implementation-defined constant
	|	*value*	Core values
	|	$\texttt{constrained}\,(\overline{mem\_iv\_c_i, pval_i}^{\,i})$	constrained value
	|	$\texttt{error}\,(string, pval)$	impl-defined static error
	|	$ctor\_val(\overline{pval_i}^{\,i})$	data constructor application
	|	$(\texttt{struct}\,ident)\{\overline{.\,member_i = pval_i}^{\,i}\}$	C struct expression
	|	$(\texttt{union}\,ident)\{.\,member = pval\}$	C union expression
pexpr	::=		pure expressions
	|	*pval*	pure values
	|	$ctor\_expr(\overline{pval_i}^{\,i})$	data constructor application
	|	$\texttt{array\_shift}\,(pval_1, \tau, pval_2)$	pointer array shift
	|	$\texttt{member\_shift}\,(pval, ident, member)$	pointer struct/union member shift
	|	$\texttt{not}\,(pval)$	boolean not
	|	$pval_1\ binop\ pval_2$	binary operations
	|	$\texttt{memberof}\,(ident, member, pval)$	C struct/union member access
	|	$name(pval_1, \ldots, pval_n)$	pure function call
	|	$\texttt{assert\_undef}\,(pval,\ UB\_name)$	
	|	$\texttt{bool\_to\_integer}\,(pval)$	

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

$bool$, _	::=		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
	|	$pval_1 == pval_2$	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` $(pval)$	
tval	::=		(effectful) top-level values
	|	`done` $pval$	end of top-level expression
	|	`undef` *UB_name*	undefined behaviour
seq_expr	::=		sequential (effectful) expressions
	|	$pval$	pure values
	|	`ccall` $(\tau, pval, \overline{pval_i}^{\,i})$	C function call
	|	`pcall` $(name, \overline{pval_i}^{\,i})$	procedure call

seq_texpr	::=		sequential top-level (effectful) expressions
	|	$tval$	(effectful) top-level values
	|	$\texttt{run}\,ident\,pval_1,\,..\,,pval_n$	run from label
	|	$\texttt{nd}\,(pval_1,\,..\,,pval_n)$	nondeterministic choice
	|	$\texttt{let}\,ident\_or\_pattern = seq\_expr\,\texttt{in}\,texpr$	pure sequencing
	|	$\texttt{case}\,pval\,\texttt{with}\,\overline{\mid pattern_i \Rightarrow texpr_i}^{\,i}\,\texttt{end}$	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
	|	$\texttt{unseq}\,(texpr_1,\,..\,,texpr_n)$	unsequenced expressions
is_texpr	::=		indet seq top-level (effectful) expressions
	|	$\texttt{let}\,\texttt{weak}\,pattern = is\_expr\,\texttt{in}\,\texttt{mu\_texpr\_aux}$	weak sequencing
	|	$\texttt{let}\,\texttt{strong}\,ident\_or\_pattern = is\_expr\,\texttt{in}\,\texttt{mu\_texpr\_aux}$	strong sequencing
texpr	::=		top-level (effectful) expressions
	|	seq_texpr	sequential (effectful) expressions
	|	is_texpr	indet seq (effectful) expressions
terminals	::=		
	|	λ	
	|	$\longrightarrow$	
	|	$\rightarrow$	
	|	$\leadsto$	
	|	$\Rightarrow$	
	|	$\Leftarrow$	
	|	$\vdash$	

$$
\begin{array}{ll}
\mid & \in \\
\mid & \Pi \\
\mid & \forall \\
\mid & \multimap \\
\mid & \supset \\
\mid & \Sigma \\
\mid & \exists \\
\mid & \star \\
\mid & \times \\
\mid & \wedge \\
\mid & \bigwedge \\
\mid & \neg \\
\mid & = \\
\mid & \neq \\
\mid & \leq \\
\mid & \geq \\
\mid & \ \& \\
\mid & . \\
\mid & | \\
\mid & +_{\mathrm{ptr}} \\
\mid & \mapsto \\
\mid & * \\
\mid & :: \\
\mid & \checkmark \\
\mid & : \\
\mid & . \\
\mid & . \\
\mid & \gg \\
\mid & ::
\end{array}
$$

	$\mid$	$\hat{}$		
	$\mid$	$\bigvee$		
z	$::=$			OCaml arbitrary-width integer
	$\mid$	i	M	literal integer
	$\mid$	$\mathrm{to\_int}(mem\_int)$	M	
	$\mid$	$\mathrm{size\_of}(\tau)$	M	size of a C type
	$\mid$	$\mathrm{offset\_of}_{tag}(member)$	M	offset of a struct member
	$\mid$	`ptr_size`	M	size of a pointer
$\mathbb{Q}$	$::=$			OCaml type for rational numbers
	$\mid$	$\frac{int_1}{int_2}$		
lit	$::=$			
	$\mid$	$ident$		
	$\mid$	`unit`		
	$\mid$	$bool$		
	$\mid$	z		
	$\mid$	$\mathbb{Q}$		
$bool\_op$	$::=$			
	$\mid$	$\neg\, term$		
	$\mid$	$term_1 = term_2$		
	$\mid$	$\bigwedge(\,\overline{term_i}^{\,i}\,)$		
	$\mid$	$\bigvee(\,\overline{term_i}^{\,i}\,)$		
	$\mid$	$term_1\ binop_{bool}\ term_2$	M	
$arith\_op$	$::=$			
	$\mid$	$term_1 + term_2$		
	$\mid$	$term_1 - term_2$		

	$\vert$	$term_1 \times term_2$		
	$\vert$	$term_1 / term_2$		
	$\vert$	$term_1\ \texttt{rem\_t}\ term_2$		
	$\vert$	$term_1\ \texttt{rem\_f}\ term_2$		
	$\vert$	$term_1\ \texttt{\^{}}\ term_2$		
	$\vert$	$term_1\ binop_{arith}\ term_2$	M	
cmp_op	$::=$			
	$\vert$	$term_1 < term_2$		less than
	$\vert$	$term_1 \leq term_2$		less than or equal
	$\vert$	$term_1\ binop_{rel}\ term_2$	M	
$list\_op$	$::=$			
	$\vert$	$\texttt{nil}$		
	$\vert$	$\texttt{tl}\ term$		
	$\vert$	$term^{(int)}$		
$tuple\_op$	$::=$			
	$\vert$	$(\ \overline{term_i}^{\,i}\ )$		
	$\vert$	$term^{(int)}$		
$pointer\_op$	$::=$			
	$\vert$	$\texttt{of\_mem\_ptr}\ mem\_ptr$		
	$\vert$	$term_1 +_{\mathrm{ptr}} term_2$		
$option\_op$	$::=$			
	$\vert$	$\texttt{none}\ BT\_t$		
	$\vert$	$\texttt{some}\ term$		
$array\_op$	$::=$			

	$\vert$	$term_1[term_2]$		
$param\_op$	$::=$			
	$\vert$	$term(term_1, .., term_n)$		
$struct\_op$	$::=$			
	$\vert$	$term.member$		
ct_pred	$::=$			
	$\vert$	$\texttt{representable}\,(\tau, term)$		
	$\vert$	$\texttt{alignedI}\,(term_1, term_2)$		
$term,\ \_$	$::=$			
	$\vert$	lit		
	$\vert$	$arith\_op$		
	$\vert$	$bool\_op$		
	$\vert$	cmp_op		
	$\vert$	$tuple\_op$		
	$\vert$	$struct\_op$		
	$\vert$	$pointer\_op$		
	$\vert$	$list\_op$		
	$\vert$	$array\_op$		
	$\vert$	ct_pred		
	$\vert$	$option\_op$		
	$\vert$	$param\_op$		
	$\vert$	$(term)$	S	parentheses
	$\vert$	$[term_1/ident]term_2$	M	substitute $term_1$ for $ident$ in $term_2$
	$\vert$	$pval$	M	only the ones which can be embeded into the SMT value grammar, so no array literals
	$\vert$	$resource$		

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
	|	$\times$	uninitalised
predicate	::=		arbitrary predicate
	|	$terms_1 \, @\!\overset{init}{\mapsto}_{predicate\_name} \, terms_2$	
resource	::=		
	|	*predicate*	
spine_elem	::=		spine element
	|	*pval*	pure value
	|	*logical_val*	logical variable
	|	*resource*	resource
arg	::=		argument types
	|	$\Pi\, ident{:}\beta.\ arg$	
	|	$\forall\, ident{:}\beta.\ arg$	
	|	$resource \multimap arg$	
	|	$term \supset arg$	
	|	*ret*	
	|	$[spine\_elem/ident]arg$ M	

ret, $\_$	$::=$			return types
	$\vert$	$\Sigma\, ident{:}\beta.\ ret$		
	$\vert$	$\exists\, ident{:}\beta.\ ret$		
	$\vert$	$resource \star ret$		
	$\vert$	$term \wedge ret$		
	$\vert$	$\mathtt{I}$		
$\mathcal{C}$	$::=$			computational var env
	$\vert$	$\cdot$		
	$\vert$	$\mathcal{C}, ident{:}BT\_t$		
	$\vert$	$\mathcal{C}, \mathcal{C}'$		
	$\vert$	$\mathrm{fresh}(\mathcal{C})$	M	identical context except with fresh variable names
$\mathcal{L}$	$::=$			logical var env
	$\vert$	$\cdot$		
	$\vert$	$\mathcal{L}, ident$		
Φ	$::=$			constraints env
	$\vert$	$\cdot$		
	$\vert$	$\Phi, term$		
$\mathcal{R}$	$::=$			resources env
	$\vert$	$\cdot$		
	$\vert$	$\mathcal{R}, resource$		
$formula$	$::=$			
	$\vert$	$judgement$		
	$\vert$	$\mathtt{smt}\,(\Phi \Rightarrow term)$		
	$\vert$	$ident{:}\beta\ \in \mathcal{C}$		
	$\vert$	$ident{:}\,\mathtt{struct}\, tag\ \&\, \overline{member_i{:}\tau_i}^{\,i}\ \in \mathtt{Globals}$		

	$\mid$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{mem\_val}_i \Rightarrow \texttt{mem}\, y_i{:}\beta_i.\, \mathit{term}_i}^{i}$ dependent on memory object model
	$\mid$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{pval}_i \Rightarrow \mathit{ident}_i{:}\beta_i.\, \mathit{term}_i}^{i}$
	$\mid$	$\overline{\mathit{pattern}_i{:}\beta_i \rightsquigarrow \mathcal{C}_i}^{i}$
	$\mid$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{tpexpr}_i \Leftarrow y_i{:}\beta_i.\, \mathit{term}_i}^{i}$
	$\mid$	$\mathcal{L} \vdash \mathit{logical\_val}{:}\beta$
object_value_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{object\_value} \Rightarrow \texttt{obj}\, \mathit{ident}{:}\beta.\, \mathit{term}$
pval_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term}$
spine_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \overline{\mathit{spine\_elem}_i}^{i} :: \mathit{arg} \gg \mathit{ret}$
pexpr_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\, \mathit{term}$
pattern_jtype	::=	
	$\mid$	$\mathit{pattern}{:}\beta \rightsquigarrow \mathcal{C}$
	$\mid$	$\mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}$
tpval_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term}$
tpexpr_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\, \mathit{term}$
action_jtype	::=	
	$\mid$	$\mathcal{C}; \mathcal{L}; \Phi; \mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret}$

$$
\begin{array}{lll}
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{seq\_expr} \Rightarrow \mathit{ret} \\
 & | & \mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{is\_expr} \Rightarrow \mathit{ret} \\
 & & \\
\mathit{judgement} & ::= & \\
 & | & \mathit{object\_value\_jtype} \\
 & | & \mathit{pval\_jtype} \\
 & | & \mathit{spine\_jtype} \\
 & | & \mathit{pexpr\_jtype} \\
 & | & \mathit{pattern\_jtype} \\
 & | & \mathit{tpval\_jtype} \\
 & | & \mathit{tpexpr\_jtype} \\
 & | & \mathit{action\_jtype} \\
 & & \\
\mathit{user\_syntax} & ::= & \\
 & | & \mathit{ident} \\
 & | & n \\
 & | & \mathit{impl\_const} \\
 & | & \mathit{mem\_int} \\
 & | & \mathit{member} \\
 & | & \\
 & | & \mathit{nat} \\
 & | & \mathit{mem\_ptr} \\
 & | & \mathit{mem\_val} \\
 & | & \\
 & | & \mathit{mem\_iv\_c} \\
 & | & \mathit{UB\_name} \\
 & | & \mathit{string} \\
 & | & \\
 & | & \\
 & | & \mathit{mem\_order}
\end{array}
$$

$|\quad linux\_mem\_order$

$|$

$|\quad logical\_val$

$|\quad Sctypes\_t$

$|\quad tag$

$|\quad \beta$

$|\quad binop$

$|\quad binop_{arith}$

$|\quad binop_{rel}$

$|\quad binop_{bool}$

$|\quad ident$

$|\quad \tau$

$|\quad ident$

$|\quad object\_value$

$|\quad loaded\_value$

$|\quad \beta$

$|\quad value$

$|\quad ctor\_val$

$|\quad ctor\_expr$

$|\quad \tau$

$|\quad name$

$|\quad pval$

$|\quad pval$

$|\quad pexpr$

$|\quad pexpr$

$|\quad tpval$

$|\quad tpval$

$|\quad ident\_opt\_\beta$

$|\quad pattern$

| $pattern$
| $ident\_or\_pattern$
| $tpexpr$
| $tpexpr$
| m_kill_kind
| $bool$
| int
| mem_action
| mem_action
| $polarity$
| pol_mem_action
| mem_op
| $tval$
| $tval$
| seq_expr
| seq_expr
| seq_texpr
| seq_texpr
| is_expr
| is_expr
| is_texpr
| is_texpr
| $texpr$
| $terminals$
| z
| $\mathbb{Q}$
| lit
| $bool\_op$
| $arith\_op$

$$
\begin{array}{ll}
| & \mathit{cmp\_op} \\
| & \mathit{list\_op} \\
| & \mathit{tuple\_op} \\
| & \mathit{pointer\_op} \\
| & \mathit{BT\_t} \\
| & \mathit{option\_op} \\
| & \mathit{array\_op} \\
| & \mathit{param\_op} \\
| & \mathit{struct\_op} \\
| & \mathit{ct\_pred} \\
| & \mathit{term} \\
| & \mathit{term} \\
| & \mathit{term} \\
| & \mathit{terms} \\
| & \mathit{predicate\_name} \\
| & \mathit{init} \\
| & \mathit{predicate} \\
| & \mathit{resource} \\
| & \mathit{spine\_elem} \\
| & \mathit{arg} \\
| & \mathit{ret} \\
| & \mathcal{C} \\
| & \mathcal{L} \\
| & \Phi \\
| & \mathcal{R} \\
| & \mathit{formula}
\end{array}
$$

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

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{mem\_int} \Rightarrow \texttt{obj}\, y{:}\texttt{integer}.\; y = \mathrm{to\_int}(\mathit{mem\_int})} \quad \textsc{Pval\_Obj\_Int}$$

$$\frac{}{\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{mem\_ptr} \Rightarrow \texttt{obj}\, y{:}\texttt{loc}.\; y = \texttt{of\_mem\_ptr}\, \mathit{mem\_ptr}} \quad \textsc{Pval\_Obj\_Ptr}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{loaded\_value}_i \Rightarrow y_i{:}\beta.\ \mathit{term}_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{array}\,(\,\overline{\mathit{loaded\_value}_i}^{\,i}\,) \Rightarrow \texttt{obj}\ y{:}\texttt{array}\,\beta.\ \bigwedge(\,\overline{[y[i]/y_i]\mathit{term}_i}^{\,i}\,)}\ \textsc{Pval\_Obj\_Arr}$$

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

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

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

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

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

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{True} \Rightarrow y{:}\texttt{bool}.\ y = \texttt{true}}\ \textsc{Pval\_True}$$

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{False} \Rightarrow y{:}\texttt{bool}.\ y = \texttt{false}}\ \textsc{Pval\_False}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{value}_i \Rightarrow y_i{:}\beta.\ \mathit{term}_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \beta[\,\overline{\mathit{value}_i}^{\,i}\,] \Rightarrow y{:}[\beta].\ \bigwedge(\,\overline{[y^{(i)}/y_i]\mathit{term}_i}^{\,i}\,)}\ \textsc{Pval\_List}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash value_i \Rightarrow y_i{:}\beta_i.\ term_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash (\ \overline{value_i}^{\,i}\ ) \Rightarrow y{:}\overline{\beta_i}^{\,i}.\ \bigwedge(\ \overline{[y^{(i)}/y_i]term_i}^{\,i}\ )}\ \textsc{Pval\_Tuple}$$

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

$$\frac{}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Nil}\,\beta(\,) \Rightarrow y{:}[\beta].\ y = \texttt{nil}}\ \textsc{Pval\_Ctor\_Nil}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow y_1{:}\beta.\ term_1\\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow y_2{:}[\beta].\ term_2\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Cons}(pval_1, pval_2) \Rightarrow y{:}[\beta].\ [y^{(1)}/y_1]term_1 \wedge [\texttt{tl}\ y/y_2]term_2}\ \textsc{Pval\_Ctor\_Cons}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow y_i{:}\beta_i.\ term_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Tuple}(\ \overline{pval_i}^{\,i}\ ) \Rightarrow y{:}\overline{\beta_i}^{\,i}.\ \bigwedge(\ \overline{[y^{(i)}/y_i]term_i}^{\,i}\ )}\ \textsc{Pval\_Ctor\_Tuple}$$

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow y_i{:}\beta.\ term_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Array}(\ \overline{pval_i}^{\,i}\ ) \Rightarrow y{:}\texttt{array}\,\beta.\ \bigwedge(\ \overline{[y[i]/y_i]term_i}^{\,i}\ )}\ \textsc{Pval\_Ctor\_Array}$$

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

$$\frac{\overline{\mathcal{C};\mathcal{L};\Phi \vdash pval_i \Rightarrow y_i{:}\beta_i.\ term_i}^{\,i}}{\mathcal{C};\mathcal{L};\Phi \vdash (\texttt{struct}\,tag)\{\ \overline{.\,member_i = pval_i}^{\,i}\ \} \Rightarrow y{:}\texttt{struct}\,tag.\ \bigwedge(\ \overline{[y.member_i/y_i]term_i}^{\,i}\ )}\ \textsc{Pval\_Struct}$$

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

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

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

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

$$\frac{\begin{array}{l}\texttt{smt}\,(\Phi\Rightarrow\mathit{resource}=\mathit{resource}')\\ \mathcal{C};\mathcal{L};\Phi;\mathcal{R}\vdash\ \overline{\mathit{spine\_elem}_i}^{\,i}\ ::\mathit{arg}\gg\mathit{ret}\end{array}}{\mathcal{C};\mathcal{L};\Phi;\mathcal{R},\mathit{resource}\vdash\mathit{resource}',\ \overline{\mathit{spine\_elem}_i}^{\,i}\ ::\mathit{resource}'\multimap\mathit{arg}\gg\mathit{ret}}\quad\textsc{Spine\_Resource}$$

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{name} \Rightarrow \_{:}\beta_1,\ \ldots,\beta_n \to \beta'.\ \_ \\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \_1{:}\beta_1.\ \_1 \ldots \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_n \Rightarrow \_n{:}\beta_n.\ \_n\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{name}(\mathit{pval}_1,\ \ldots,\mathit{pval}_n) \Rightarrow y{:}\beta'.\ \mathit{term}'}\ \textsc{PExpr\_Call}$$

$\boxed{\mathit{pattern}{:}\beta \rightsquigarrow \mathcal{C}}$

$\boxed{\mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}}$

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pexpr} \Rightarrow \_{:}\beta.\ \_ \\ \mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}' \\ \mathcal{C}, \mathrm{fresh}(\mathcal{C}');\mathcal{L};\Phi \vdash [\mathrm{fresh}(\mathcal{C}')/\mathcal{C}']\mathit{tpexpr} \Leftarrow y{:}\beta.\ \mathit{term}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathtt{let}\,\mathit{ident\_or\_pattern} = \mathit{pexpr}\,\mathtt{in}\,\mathit{tpexpr} \Leftarrow y{:}\beta.\ \mathit{term}} \quad \textsc{TPExpr\_Let}$$

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \_{:}\beta.\ \_ \\ \overline{\mathit{pattern}_i{:}\beta \rightsquigarrow \mathcal{C}_i}^{\,i} \\ \overline{\mathcal{C}, \mathrm{fresh}(\mathcal{C}_i);\mathcal{L};\Phi \vdash [\mathrm{fresh}(\mathcal{C}_i)/\mathcal{C}_i]\mathit{tpexpr}_i \Leftarrow y{:}\beta.\ \mathit{term}}^{\,i}\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathtt{case}\,\mathit{pval}\,\mathtt{of}\,\overline{\mid \mathit{pattern}_i \Rightarrow \mathit{tpexpr}_i}^{\,i}\,\mathtt{end} \Leftarrow y{:}\beta.\ \mathit{term}} \quad \textsc{TPExpr\_Case}$$

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \_\mathtt{:loc}.\ \_\\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}' \Rightarrow y'{:}\beta_\tau.\ \mathit{term}'\\ \mathtt{smt}\,(\Phi, \mathit{term}' \Rightarrow \mathtt{representable}\,(\tau, y'))\\ \mathtt{smt}\,(\Phi \Rightarrow \mathit{pval}_0 = \mathit{pval}_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,[\mathit{pval}_0]\ 1{\mapsto_\tau}\ [\_'']\vdash \mathtt{store}\,(\_,\tau,\mathit{pval}_1,\mathit{pval}',\_)\Rightarrow \Sigma\,\_\mathtt{:unit}.\ \exists\, y'{:}\beta_\tau.\ \mathit{term}' \wedge [\mathit{pval}_0]\ 1\overset{\checkmark}{\mapsto}_\tau\ [y'] \star \mathtt{I}} \quad \textsc{Action\_Store}$$

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow \_\mathtt{:loc}.\ \_\\ \mathtt{smt}\,(\Phi \Rightarrow \mathit{pval}_0 = \mathit{pval}_1)\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,[\mathit{pval}_0]\ 1{\mapsto_\tau}\ [\_']\vdash \mathtt{kill}\,(\mathtt{static}\,\tau,\mathit{pval}_1)\Rightarrow \Sigma\,\_\mathtt{:unit}.\ \mathtt{I}} \quad \textsc{Action\_Kill\_Static}$$

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

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

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

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

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

```
Definition rules:        43 good    0 bad
Definition rule clauses: 102 good    0 bad
```