$ident$, x, y, y_p, _	v is for values, subscript p is for pointers, rest are identifiers
$impl\_const$	implementation-defined constant
mem_int	memory integer value
$member$	C struct/union member name
	Ott-hack, ignore (annotations)
nat	OCaml arbitrary-width natural number
n, i	index variables
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
k	OCaml fixed-width integer
	Ott-hack, ignore (OCaml type variable bt)

Sctypes_t, τ	::=			C type
	|	$\tau *$		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
	|	$[\beta]$		list
	|	$\beta_1 \times \,..\, \times \beta_n$		tuple
	|	`struct` *tag*		struct
	|	$\{\beta\}$		set
	|	`opt` (β)		option
	|	$\beta_1, .., \beta_n \to \beta$		parameter types
	|	of_ctype(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
$\mathit{object\_value}$	::=		C object values (inhabitants of object types), which can be read/stored
	|	$\mathit{mem\_int}$	integer value
	|	$\mathit{mem\_ptr}$	pointer value
	|	$\texttt{array}\,(\,\overline{\mathit{loaded\_value}_i}^{\,i}\,)$	C array value
	|	$(\,\texttt{struct}\,\mathit{ident})\{\,\overline{.\,\mathit{member}_i{:}\tau_i = \mathit{mem\_val}_i}^{\,i}\,\}$	C struct value
	|	$(\,\texttt{union}\,\mathit{ident})\{\,.\,\mathit{member} = \mathit{mem\_val}\}$	C union value
$\mathit{loaded\_value}$	::=		potentially unspecified C object values
	|	$\texttt{specified}\,\mathit{object\_value}$	specified loaded value
value	::=		Core values
	|	$\mathit{object\_value}$	C object value
	|	$\mathit{loaded\_value}$	loaded C object value
	|	`Unit`	unit
	|	`True`	boolean true
	|	`False`	boolean false
	|	$\beta[\mathit{value}_1, \,..\,, \mathit{value}_i]$	list
	|	$(\mathit{value}_1, \,..\,, \mathit{value}_i)$	tuple
ctor	::=		data constructors
	|	$\texttt{Nil}\,\beta$	empty list
	|	`Cons`	list cons
	|	`Tuple`	tuple

	|	`Array`	C array
	|	`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
	|	`Specified`	non-unspecified loaded value
	|	`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
	|	$ctor(\overline{pval_i}^{\,i})$	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
	|	$\texttt{array\_shift}\,(pval_1, \tau, pval_2)$	pointer array shift

	$\mid$	$\texttt{member\_shift}\,(\mathit{pval}, \mathit{ident}, \mathit{member})$	pointer struct/union member shift
	$\mid$	$\texttt{not}\,(\mathit{pval})$	boolean not
	$\mid$	$\mathit{pval}_1\ \mathit{binop}\ \mathit{pval}_2$	binary operations
	$\mid$	$\texttt{memberof}\,(\mathit{ident}, \mathit{member}, \mathit{pval})$	C struct/union member access
	$\mid$	$\mathit{name}(\mathit{pval}_1, \,..\,, \mathit{pval}_n)$	pure function call
	$\mid$	$\texttt{assert\_undef}\,(\mathit{pval},\ \mathit{UB\_name})$	
	$\mid$	$\texttt{bool\_to\_integer}\,(\mathit{pval})$	
	$\mid$	$\texttt{conv\_int}\,(\tau, \mathit{pval})$	
	$\mid$	$\texttt{wrapI}\,(\tau, \mathit{pval})$	
tpval	$::=$		top-level pure values
	$\mid$	$\texttt{undef}\ \mathit{UB\_name}$	undefined behaviour
	$\mid$	$\texttt{done}\,\mathit{pval}$	pure done
$\mathit{ident\_opt\_\beta}$	$::=$		type annotated optional identifier
	$\mid$	$\_{:}\beta$	
	$\mid$	$\mathit{ident}{:}\beta$	
$\mathit{pattern}$	$::=$		
	$\mid$	$\mathit{ident\_opt\_\beta}$	
	$\mid$	$\mathit{ctor}(\,\overline{\mathit{pattern}_i}^{\,i}\,)$	
$\mathit{ident\_or\_pattern}$	$::=$		
	$\mid$	ident	
	$\mid$	$\mathit{pattern}$	
tpexpr	$::=$		top-level pure expressions
	$\mid$	tpval	top-level pure values
	$\mid$	$\texttt{case}\,\mathit{pval}\,\texttt{of}\,\overline{\mid \mathit{pattern}_i \Rightarrow \mathit{tpexpr}_i}^{\,i}\ \texttt{end}$	pattern matching
	$\mid$	$\texttt{let}\,\mathit{ident\_or\_pattern} = \mathit{pexpr}\,\texttt{in}\,\mathit{tpexpr}$	pure let

	|	`if` $pval$ `then` $tpexpr_1$ `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	::=			
	|	`dynamic`		
	|	`static` τ		
$bool$, _	::=			OCaml booleans
	|	`true`		
	|	`false`		
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

	$\vert$	$\texttt{ccall}\,(\tau, \mathit{pval}, \overline{\mathit{pval}_i}^{\,i}\,)$	C function call
	$\vert$	$\texttt{pcall}\,(\mathit{name}, \overline{\mathit{pval}_i}^{\,i}\,)$	procedure call
$\mathit{seq\_texpr}$	$::=$		sequential top-level (effectful) expressions
	$\vert$	tval	(effectful) top-level values
	$\vert$	$\texttt{run}\,\mathit{ident}\,\mathit{pval}_1, \,..\,, \mathit{pval}_n$	run from label
	$\vert$	$\texttt{nd}\,(\mathit{pval}_1, \,..\,, \mathit{pval}_n)$	nondeterministic choice
	$\vert$	$\texttt{let}\,\mathit{ident\_or\_pattern} = \mathit{seq\_expr}\,\texttt{in}\,\mathit{texpr}$	pure sequencing
	$\vert$	$\texttt{case}\,\mathit{pval}\,\texttt{with}\,\overline{\vert\ \mathit{pattern}_i \Rightarrow \mathit{texpr}_i}^{\,i}\,\texttt{end}$	pattern matching
	$\vert$	$\texttt{if}\,\mathit{pval}\,\texttt{then}\,\mathit{texpr}_1\,\texttt{else}\,\mathit{texpr}_2$	conditional
	$\vert$	$\texttt{bound}\,[k](\mathit{is\_texpr})$	limit scope of indet seq behaviour, absent at runtime
$\mathit{is\_expr}$	$::=$		indet seq (effectful) expressions
	$\vert$	$\texttt{memop}\,(\mathit{mem\_op})$	pointer op involving memory
	$\vert$	$\mathit{pol\_mem\_action}$	memory action
	$\vert$	$\texttt{unseq}\,(\mathit{texpr}_1, \,..\,, \mathit{texpr}_n)$	unsequenced expressions
$\mathit{is\_texpr}$	$::=$		indet seq top-level (effectful) expressions
	$\vert$	$\texttt{let}\,\texttt{weak}\,\mathit{pattern} = \mathit{is\_expr}\,\texttt{in}\,\texttt{mu\_texpr\_aux}$	weak sequencing
	$\vert$	$\texttt{let}\,\texttt{strong}\,\mathit{ident\_or\_pattern} = \mathit{is\_expr}\,\texttt{in}\,\texttt{mu\_texpr\_aux}$	strong sequencing
texpr	$::=$		top-level (effectful) expressions
	$\vert$	$\mathit{seq\_texpr}$	sequential (effectful) expressions
	$\vert$	$\mathit{is\_texpr}$	indet seq (effectful) expressions
$\mathit{terminals}$	$::=$		
	$\vert$	λ	
	$\vert$	$\longrightarrow$	
	$\vert$	$\rightarrow$	
	$\vert$	$\leadsto$	

$|\quad \Rightarrow$

$|\quad \Leftarrow$

$|\quad \vdash$

$|\quad \in$

$|\quad \Pi$

$|\quad \forall$

$|\quad \multimap$

$|\quad \supset$

$|\quad \Sigma$

$|\quad \exists$

$|\quad \star$

$|\quad \times$

$|\quad \wedge$

$|\quad \bigwedge$

$|\quad \neg$

$|\quad =$

$|\quad \neq$

$|\quad \leq$

$|\quad \geq$

$|\quad \&$

$|\quad \cdot$

$|\quad |$

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

$|\quad \mapsto$

$|\quad *$

$|\quad ::$

$|\quad \checkmark$

$|\quad :$

$|\quad .$

	$\vert$	.		
z	$::=$			OCaml arbitrary-width integer
	$\vert$	$\mathrm{of\_mem\_int}(\mathit{mem\_int})$	M	
	$\vert$	$\mathrm{of\_ctype}(\tau)$	M	size of a C type
	$\vert$	$\texttt{ptr\_size}$	M	size of a pointer
$\mathbb{Q}$	$::=$			OCaml type for rational numbers
	$\vert$	$\frac{k_1}{k_2}$		
lit	$::=$			
	$\vert$	ident		
	$\vert$	$\texttt{unit}$		
	$\vert$	bool		
	$\vert$	$\texttt{int}\, z$		
	$\vert$	$\mathbb{Q}$		
	$\vert$	$\texttt{ptr}\, z$		
$\mathit{bool\_op}$	$::=$			
	$\vert$	$\neg\, \mathit{term}$		
	$\vert$	$\mathit{term}_1 = \mathit{term}_2$		
	$\vert$	$\bigwedge(\,\overline{\mathit{term}_i}^{\,i}\,)$		
	$\vert$	$\mathit{term}_1 \wedge \mathit{term}_2$	M	
$\mathit{arith\_op}$	$::=$			
	$\vert$	$\mathit{term}_1 \times \mathit{term}_2$		
$\mathit{list\_op}$	$::=$			
	$\vert$	$\texttt{nil}$		
	$\vert$	$\mathit{term}_1 :: \mathit{term}_2$		

	$\mid$	$\texttt{tl}\, term$
	$\mid$	$[term_1, \, .. \, , term_n]$
	$\mid$	$term^{(k)}$
$tuple\_op$	$::=$	
	$\mid$	$(term_1, \, .. \, , term_n)$
	$\mid$	$term^{(k)}$
$pointer\_op$	$::=$	
	$\mid$	$\texttt{of\_mem\_ptr}\, mem\_ptr$
	$\mid$	$term_1 +_{\mathrm{ptr}} term_2$
$option\_op$	$::=$	
	$\mid$	$\texttt{none}\, BT\_t$
	$\mid$	$\texttt{some}\, term$
$array\_op$	$::=$	
	$\mid$	$term_1[term_2]$
$param\_op$	$::=$	
	$\mid$	$term(term_1, \, .. \, , term_n)$
$struct\_op$	$::=$	
	$\mid$	$term.member$
ct_pred	$::=$	
	$\mid$	$\texttt{representable}\, (\tau, term)$
	$\mid$	$\texttt{alignedI}\, (term_1, term_2)$
$term, \, \_$	$::=$	

	$\vert$	lit		
	$\vert$	$arith\_op$		
	$\vert$	$bool\_op$		
	$\vert$	$tuple\_op$		
	$\vert$	$struct\_op$		
	$\vert$	$pointer\_op$		
	$\vert$	$list\_op$		
	$\vert$	$array\_op$		
	$\vert$	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 $term_2$
	$\vert$	$\texttt{to\_term}\, pval$	M	
$terms$	::=			non-empty list of terms
	$\vert$	$[term]$		
	$\vert$	$[term, \ldots]$		
$predicate\_name$	::=			names of predicates
	$\vert$	$Sctypes\_t$		C type
	$\vert$	$string$		arbitrary
$init$,	::=			initialisation status
	$\vert$	$\checkmark$		initialised
	$\vert$	$\times$		uninitalised
$predicate$	::=			arbitrary predicate
	$\vert$	$terms_1 \, \mathbb{Q}\overset{init}{\mapsto}_{predicate\_name} \, terms_2$		

$\mathit{resource}$	$::=$			
	$\vert$	$\mathit{predicate}$		
arg	$::=$			argument types
	$\vert$	$\Pi\,\mathit{ident}{:}\beta.\ \mathit{arg}$		
	$\vert$	$\forall\,\mathit{ident}{:}\beta.\ \mathit{arg}$		
	$\vert$	$\mathit{resource} \multimap \mathit{arg}$		
	$\vert$	$\mathit{term} \supset \mathit{arg}$		
	$\vert$	$\mathtt{I}$		
ret, _	$::=$			return types
	$\vert$	$\Sigma\,\mathit{ident}{:}\beta.\ \mathit{ret}$		
	$\vert$	$\exists\,\mathit{ident}{:}\beta.\ \mathit{ret}$		
	$\vert$	$\mathit{resource} \star \mathit{ret}$		
	$\vert$	$\mathit{term} \wedge \mathit{ret}$		
	$\vert$	$\mathtt{I}$		
$\mathcal{C}$	$::=$			computational var env
	$\vert$	$\cdot$		
	$\vert$	$\mathcal{C}, \mathit{ident}{:}\mathit{BT\_t}$		
	$\vert$	$\mathcal{C}, \mathcal{C}'$		
	$\vert$	$\mathrm{fresh}(\mathcal{C})$	M	identical context except with fresh variable names
$\mathcal{L}$	$::=$			logical var env
	$\vert$	$\cdot$		
	$\vert$	$\mathcal{L}, \mathit{ident}$		
Φ	$::=$			constraints env
	$\vert$	$\cdot$		
	$\vert$	Φ, term		

$\mathcal{R}$	::=		resources env
	$\vert$	$\cdot$	
	$\vert$	$\mathcal{R}, \mathit{resource}$	
formula	::=		
	$\vert$	$\mathit{judgement}$	
	$\vert$	$\mathtt{smt}\,(\Phi \Rightarrow \mathit{term})$	
	$\vert$	$\mathit{ident}{:}\beta\ \in \mathcal{C}$	
	$\vert$	$\mathit{ident}{:}\,\mathtt{struct}\,\mathit{tag}\ \&\ \overline{\mathit{member}_i{:}\tau_i}^{\,i}\ \in \mathtt{Globals}$	
	$\vert$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{loaded\_value}_i \Rightarrow y_i{:}\beta_i.\ \mathit{term}_i}^{\,i}$	specified implicitly
	$\vert$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{mem\_val}_i \Rightarrow \mathtt{mem}\ y_i{:}\beta_i.\ \mathit{term}_i}^{\,i}$	dependent on memory object model
	$\vert$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{value}_i \Rightarrow y_i{:}\beta_i.\ \mathit{term}_i}^{\,i}$	specified implicitly
	$\vert$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{pval}_i \Rightarrow \mathit{ident}_i{:}\beta_i.\ \mathit{term}_i}^{\,i}$	
	$\vert$	$\overline{\mathit{pattern}_i{:}\beta_i \rightsquigarrow \mathcal{C}_i}^{\,i}$	
	$\vert$	$\overline{\mathcal{C}_i; \mathcal{L}_i; \Phi_i \vdash \mathit{tpexpr}_i \Leftarrow y_i{:}\beta_i.\ \mathit{term}_i}^{\,i}$	
object_value_jtype	::=		
	$\vert$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{object\_value} \Rightarrow \mathtt{obj}\ \mathit{ident}{:}\beta.\ \mathit{term}$	
pval_jtype	::=		
	$\vert$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pval} \Rightarrow \mathit{ident}{:}\beta.\ \mathit{term}$	
pexpr_jtype	::=		
	$\vert$	$\mathcal{C}; \mathcal{L}; \Phi \vdash \mathit{pexpr} \Rightarrow \mathit{ident}{:}\beta.\ \mathit{term}$	
pattern_jtype	::=		
	$\vert$	$\mathit{pattern}{:}\beta \rightsquigarrow \mathcal{C}$	
	$\vert$	$\mathit{ident\_or\_pattern}{:}\beta \rightsquigarrow \mathcal{C}$	
tpval_jtype	::=		

	$\mid$	$\mathcal{C};\mathcal{L};\Phi \vdash \mathit{tpval} \Leftarrow \mathit{ident}{:}\beta.\ \mathit{term}$
$\mathit{tpexpr\_jtype}$	$::=$	
	$\mid$	$\mathcal{C};\mathcal{L};\Phi \vdash \mathit{tpexpr} \Leftarrow \mathit{ident}{:}\beta.\ \mathit{term}$
$\mathit{action\_jtype}$	$::=$	
	$\mid$	$\mathcal{C};\mathcal{L};\Phi;\mathcal{R} \vdash \mathit{mem\_action} \Rightarrow \mathit{ret}$
$\mathit{judgement}$	$::=$	
	$\mid$	$\mathit{object\_value\_jtype}$
	$\mid$	$\mathit{pval\_jtype}$
	$\mid$	$\mathit{pexpr\_jtype}$
	$\mid$	$\mathit{pattern\_jtype}$
	$\mid$	$\mathit{tpval\_jtype}$
	$\mid$	$\mathit{tpexpr\_jtype}$
	$\mid$	$\mathit{action\_jtype}$
$\mathit{user\_syntax}$	$::=$	
	$\mid$	ident
	$\mid$	$\mathit{impl\_const}$
	$\mid$	$\mathit{mem\_int}$
	$\mid$	member
	$\mid$	
	$\mid$	nat
	$\mid$	n
	$\mid$	$\mathit{mem\_ptr}$
	$\mid$	$\mathit{mem\_val}$
	$\mid$	
	$\mid$	$\mathit{mem\_iv\_c}$
	$\mid$	$\mathit{UB\_name}$

| $string$
|
|
| mem_order
| $linux\_mem\_order$
| k
|
| $Sctypes\_t$
| tag
| β
| $binop$
| $ident$
| τ
| $ident$
| $object\_value$
| $loaded\_value$
| β
| $value$
| $ctor$
| τ
| $name$
| $pval$
| $pval$
| $pexpr$
| $pexpr$
| $tpval$
| $tpval$
| $ident\_opt\_\beta$
| $pattern$

$|$ $pattern$
$|$ $ident\_or\_pattern$
$|$ $tpexpr$
$|$ $tpexpr$
$|$ m_kill_kind
$|$ $bool$
$|$ 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
$|$ $bool\_op$
$|$ $arith\_op$
$|$ $list\_op$

| $tuple\_op$
| $pointer\_op$
| BT_t
| $option\_op$
| $array\_op$
| $param\_op$
| $struct\_op$
| ct_pred
| $term$
| $term$
| $term$
| $\ldots$
| $terms$
| $predicate\_name$
| $init$
| $predicate$
| $resource$
| arg
| ret
| $\mathcal{C}$
| $\mathcal{L}$
| Φ
| $\mathcal{R}$
| $formula$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{value}_1 \Rightarrow y_1{:}\beta.\ \mathit{term}_1 \,..\, \mathcal{C};\mathcal{L};\Phi \vdash \mathit{value}_n \Rightarrow y_n{:}\beta.\ \mathit{term}_n}{\mathcal{C};\mathcal{L};\Phi \vdash \beta[\mathit{value}_1,\,..\,,\mathit{value}_n] \Rightarrow y{:}[\beta].\ \bigwedge([y^{(k_1)}/y_1]\mathit{term}_1,\,..\,,[y^{(k_n)}/y_n]\mathit{term}_n)}\ \textsc{Pval\_List}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash value_1 \Rightarrow y_1{:}\beta_1.\ term_1 \,..\, \mathcal{C};\mathcal{L};\Phi \vdash value_n \Rightarrow y_n{:}\beta_n.\ term_n}{\mathcal{C};\mathcal{L};\Phi \vdash (value_1,\,..\,,value_n) \Rightarrow y{:}\beta_1 \times\ ..\ \times \beta_n.\ \bigwedge([y^{(k_1)}/y_1]term_1,\,..\,,[y^{(k_n)}/y_n]term_n)} \quad \textsc{Pval\_Tuple}$$

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow y_1{:}\beta.\ term_1\\ \mathcal{C};\mathcal{L};\Phi \vdash pval_2 \Rightarrow y_2{:}[\beta].\ term_2\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Cons}(pval_1, pval_2) \Rightarrow y{:}[\beta].\ [y^{(k_1)}/y_1]term_1 \wedge [\texttt{tl}\ y/y_2]term_2} \quad \textsc{Pval\_Ctor\_Cons}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow y_1{:}\beta_1.\ term_1 \,..\, \mathcal{C};\mathcal{L};\Phi \vdash pval_n \Rightarrow y_n{:}\beta_n.\ term_n}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Tuple}(pval_1,\,..\,,pval_n) \Rightarrow y{:}\beta_1 \times\ ..\ \times \beta_n.\ \bigwedge([y^{(k_1)}/y_1]term_1,\,..\,,[y^{(k_n)}/y_n]term_n)} \quad \textsc{Pval\_Ctor\_Tuple}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash pval_1 \Rightarrow y_1{:}\beta.\ term_1 \,..\, \mathcal{C};\mathcal{L};\Phi \vdash pval_n \Rightarrow y_n{:}\beta.\ term_n}{\mathcal{C};\mathcal{L};\Phi \vdash \texttt{Array}(pval_1,\,..\,,pval_n) \Rightarrow y{:}\texttt{array}\,\beta.\ \bigwedge([y[\texttt{int}\,z_1]/y_1]term_1,\,..\,,[y[\texttt{int}\,z_n]/y_n]term_n)} \quad \textsc{Pval\_Ctor\_Array}$$

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

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

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

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

$$\frac{\begin{array}{c}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_1 \Rightarrow y_1{:}\mathtt{loc}.\ \mathit{term}_1\\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}_2 \Rightarrow y_2{:}\mathtt{integer}.\ \mathit{term}_2\end{array}}{\mathcal{C};\mathcal{L};\Phi \vdash \mathtt{array\_shift}\,(\mathit{pval}_1, \tau, \mathit{pval}_2) \Rightarrow y{:}\mathtt{loc}.\ y = y_1 +_{\text{ptr}} y_2 \times \mathtt{int}\,\text{of\_ctype}(\tau) \wedge \mathit{term}_1 \wedge \mathit{term}_2}\quad \textsc{PExpr\_Array\_Shift}$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l}\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \_{:}\texttt{loc}.\ \_\\ \mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval}' \Rightarrow \_'{:}\mathrm{of\_ctype}(\tau).\ \_'\\ \texttt{smt}\,(\Phi \Rightarrow \texttt{representable}\,(\tau, \texttt{to\_term}\,\mathit{pval}'))\end{array}}{\mathcal{C};\mathcal{L};\Phi;\cdot,[\texttt{to\_term}\,\mathit{pval}]\ 1\mapsto_\tau [\_''] \vdash \texttt{store}\,(\_,\tau,\mathit{pval},\mathit{pval}',\_) \Rightarrow \Sigma\,\_{:}\texttt{unit}.\ [\texttt{to\_term}\,\mathit{pval}]\ 1\overset{\checkmark}{\mapsto}_\tau\ [\texttt{to\_term}\,\mathit{pval}'] \star \texttt{I}}\quad \textsc{Action\_Store}$$

$$\frac{\mathcal{C};\mathcal{L};\Phi \vdash \mathit{pval} \Rightarrow \_{:}\texttt{loc}.\ \_}{\mathcal{C};\mathcal{L};\Phi;\cdot,[\texttt{to\_term}\,\mathit{pval}]\ 1\mapsto_\tau [\_'] \vdash \texttt{kill}\,(\texttt{static}\,\tau,\mathit{pval}) \Rightarrow \Sigma\,\_{:}\texttt{unit}.\ \texttt{I}}\quad \textsc{Action\_Kill\_Static}$$

```
Definition rules:        30 good    0 bad
Definition rule clauses: 66 good    0 bad
```