n, m, i, j	Index variables for meta-lists
num, $numZero$, $numOne$	Numeric literals
nat	
hex	Bit vector literal, specified by C-style hex number
bin	Bit vector literal, specified by C-style binary number
$string$	String literals
$regexp$	Regular expresions, as a string literal
$real$	Real number literal
$value$	
x, y, z	identifier
ix	infix identifier

l	::=			source location
	|			
$annot$	::=			
	|			
id_aux	::=			Identifier
	|	x		
	|	(**operator** x)		remove infix status
	|	**bool**	S	
	|	**not**	S	
	|	**atom**	S	
	|	*real*	S	
	|	*string*	S	
	|	**bitvector**	S	
	|	**bit**	S	
	|	**unit**	S	
	|	**exception**	S	
	|	**int**	S	
	|	**list**	S	
	|	**vector**	S	
	|	**register**	S	
	|	**range**	S	
id	::=			
	|	$l\ id\_aux$		
kid_aux	::=			kinded IDs: Type, Int, and Order variables
	|	$'x$		
kid	::=			
	|	$l\ kid\_aux$		
$kind\_aux$	::=			base kind
	|	**Type**		kind of types
	|	**Int**		kind of natural number size expressions
	|	**Order**		kind of vector order specifications
	|	**Bool**		kind of constraints
$kind$	::=			
	|	$l\ kind\_aux$		
$nexp\_aux$	::=			numeric expression, of kind Int
	|	id		abbreviation identifier
	|	kid		variable
	|	num		constant
	|	$id(nexp_1, \ldots, nexp_n)$		app
	|	$nexp_1 * nexp_2$		product
	|	$nexp_1 + nexp_2$		sum

	$\vert$	$nexp_1 - nexp_2$		subtraction
	$\vert$	$2\uparrow nexp$		exponential
	$\vert$	$-nexp$		unary negation
	$\vert$	$(nexp)$	S	
	$\vert$	$nexp_1 + \ldots + nexp_n$	M	
nexp	::=			
	$\vert$	*l nexp_aux*		
order_aux	::=			vector order specifications, of kind Order
	$\vert$	*kid*		variable
	$\vert$	**inc**		increasing
	$\vert$	**dec**		decreasing
	$\vert$	(*order*)	S	
order	::=			
	$\vert$	*l order_aux*		
base_effect_aux	::=			effect
	$\vert$	**rreg**		read register
	$\vert$	**wreg**		write register
	$\vert$	**rmem**		read memory
	$\vert$	**rmemt**		read memory and tag
	$\vert$	**wmem**		write memory
	$\vert$	**wmea**		signal effective address for writing memory
	$\vert$	**exmem**		determine if a store-exclusive (ARM) is going
	$\vert$	**wmv**		write memory, sending only value
	$\vert$	**wmvt**		write memory, sending only value and tag
	$\vert$	**barr**		memory barrier
	$\vert$	**depend**		dynamic footprint
	$\vert$	**undef**		undefined-instruction exception
	$\vert$	**unspec**		unspecified values
	$\vert$	**nondet**		nondeterminism, from **nondet**
	$\vert$	**escape**		potential exception
	$\vert$	**config**		configuration option
base_effect	::=			
	$\vert$	*l base_effect_aux*		
effect_aux	::=			
	$\vert$	$\{base\_effect_1, .., base\_effect_n\}$		effect set
	$\vert$	**pure**	M	sugar for empty effect set
effect	::=			
	$\vert$	*l effect_aux*		
typ_aux	::=			type expressions, of kind Type
	$\vert$			
	$\vert$	*id*		defined type

	$\|$	kid		type variable
	$\|$	$(typ_1, \dots, typ_n) \rightarrow typ_2\ \mathbf{effect}\ effect$		Function (first-order only)
	$\|$	$typ_1 \leftrightarrow typ_2\ \mathbf{effect}\ effect$		Mapping
	$\|$	$(typ_1, \dots., typ_n)$		Tuple
	$\|$	$id(typ\_arg_1, \dots, typ\_arg_n)$		type constructor application
	$\|$	(typ)	S	
	$\|$	$\{kinded\_id_1 \dots kinded\_id_n, n\_constraint.typ\}$		
	$\|$	typ_{exp}	M	
	$\|$	typ_{lexp}	M	
	$\|$	typ_{pat}	M	
	$\|$	$\sigma(typ)$	M	
typ	$::=$			
	$\|$	$l\ typ\_aux$		
typ_arg_aux	$::=$			type constructor arguments of al
	$\|$	$nexp$		
	$\|$	typ		
	$\|$	$order$		
	$\|$	$n\_constraint$		
typ_arg	$::=$			
	$\|$	$l\ typ\_arg\_aux$		
$n\_constraint\_aux$	$::=$			constraint over kind Int
	$\|$	$nexp \equiv nexp'$		
	$\|$	$nexp \geq nexp'$		
	$\|$	$nexp > nexp'$		
	$\|$	$nexp \leq nexp'$		
	$\|$	$nexp < nexp'$		
	$\|$	$nexp! = nexp'$		
	$\|$	$kid\ \mathbf{IN}\ \{num_1, \dots, num_n\}$		
	$\|$	$n\_constraint \wedge n\_constraint'$		
	$\|$	$n\_constraint \| n\_constraint'$		
	$\|$	$id(typ\_arg_0, \dots, typ\_arg_n)$		
	$\|$	kid		
	$\|$	**true**		
	$\|$	**false**		
$n\_constraint$	$::=$			
	$\|$	$l\ n\_constraint\_aux$		
$kinded\_id\_aux$	$::=$			optionally kind-annotated identif
	$\|$	$kind\ kid$		kind-annotated variable
	$\|$	kid	S	
$kinded\_id$	$::=$			
	$\|$	$l\ kinded\_id\_aux$		

$quant\_item\_aux$	::=		kinded identifier or Int constraint
	|	$kinded\_id$	optionally kinded identifier
	|	$n\_constraint$	constraint
	|	$kinded\_id_0 \ldots kinded\_id_n$	
$quant\_item$	::=		
	|	$l\ quant\_item\_aux$	
$typquant\_aux$	::=		type quantifiers and constraints
	|	$\mathbf{forall}\ quant\_item_1, \ldots, quant\_item_n.$	
	|		empty
$typquant$	::=		
	|	$l\ typquant\_aux$	
$typschm\_aux$	::=		type scheme
	|	$typquant\ typ$	
$typschm$	::=		
	|	$l\ typschm\_aux$	
$type\_def$	::=		
	|	$type\_def\_aux$	
$type\_def\_aux$	::=		type definition body
	|	$\mathbf{type}\ id\ typquant = typ\_arg$	type abbreviation
	|	$\mathbf{typedef}\ id = \mathbf{const\ struct}\ typquant\{typ_1\ id_1; \ldots; typ_n\ id_n\ ;^?\}$	struct type definition
	|	$\mathbf{typedef}\ id = \mathbf{const\ union}\ typquant\{type\_union_1; \ldots; type\_union_n\ ;^?\}$	tagged union type definition
	|	$\mathbf{typedef}\ id = \mathbf{enumerate}\ \{id_1; \ldots; id_n\ ;^?\}$	enumeration type definition
	|	$\mathbf{bitfield}\ id : typ = \{id_1 : index\_range_1, \ldots, id_n : index\_range_n\}$	register mutable bitfield type definition
$type\_union\_aux$	::=		type union constructors
	|	$typ\ id$	
$type\_union$	::=		
	|	$l\ type\_union\_aux$	
$index\_range\_aux$	::=		index specification, for bitfields in register types
	|	$nexp$	single index
	|	$nexp_1..nexp_2$	index range
	|	$index\_range_1, index\_range_2$	concatenation of index ranges
$index\_range$	::=		
	|	$l\ index\_range\_aux$	

lit_aux	::=		literal constant
	|	()	
	|	**bitzero**	
	|	**bitone**	
	|	**true**	
	|	**false**	
	|	*num*	natural number constant
	|	*hex*	bit vector constant, C-style
	|	*bin*	bit vector constant, C-style
	|	$string_1$	string constant
	|	**undefined**	undefined-value constant
	|	*real*	
lit	::=		
	|	*l lit_aux*	
$;^?$	::=		optional semi-colon
	|		
	|	;	
typ_pat_aux	::=		type pattern
	|	_	
	|	*kid*	
	|	$id(typ\_pat_1, .., typ\_pat_n)$	
typ_pat	::=		
	|	*l typ_pat_aux*	
pat_aux	::=		pattern
	|	*lit*	literal constant pattern
	|	_	wildcard
	|	$pat_1 \| pat_2$	pattern disjunction
	|	$\sim pat$	pattern negation
	|	(pat **as** id)	named pattern
	|	$(typ)pat$	typed pattern
	|	*id*	identifier
	|	*pat typ_pat*	bind pattern to type variable
	|	$id(pat_1, .., pat_n)$	union constructor pattern
	|	$[pat_1, ..., pat_n]$	vector pattern
	|	$pat_1 @ ... @ pat_n$	concatenated vector pattern
	|	$(pat_1, ..., pat_n)$	tuple pattern
	|	$[\|\|pat_1, .., pat_n\|\|]$	list pattern
	|	(pat)	S
	|	$pat_1 :: pat_2$	Cons patterns
	|	$pat_1 \uparrow\uparrow ... \uparrow\uparrow pat_n$	string append pattern, x ‘y
pat	::=		
	|	*annot pat_aux*	

loop	::=		
	|	**while**	
	|	**until**	
internal_loop_measure_aux	::=		internal syntax r
	|		
	|	**termination_measure** $\{$*exp*$\}$	
internal_loop_measure	::=		
	|	l *internal_loop_measure_aux*	
exp_aux	::=		expression
	|	$\{exp_1; \ldots; exp_n\}$	sequential blo
	|	*id*	identifier
	|	*lit*	literal constan
	|	$(typ)exp$	cast
	|	$id(exp_1, .., exp_n)$	function appl
	|	$exp_1\ id\ exp_2$	infix function
	|	$(exp_1, ..., exp_n)$	tuple
	|	**if** exp_1 **then** exp_2 **else** exp_3	conditional
	|	*loop internal_loop_measure* $exp_1\ exp_2$	
	|	**foreach** $(id$ **from** exp_1 **to** exp_2 **by** exp_3 **in** $order)exp_4$	for loop
	|	$[exp_1, \ldots, exp_n]$	vector (indexe
	|	$exp[exp']$	vector access
	|	$exp[exp_1..exp_2]$	subvector ext
	|	$[exp$ **with** $exp_1 = exp_2]$	vector functio
	|	$[exp$ **with** $exp_1..exp_2 = exp_3]$	vector subran
	|	$exp_1 @ exp_2$	vector concat
	|	$[\|exp_1, .., exp_n\|]$	list
	|	$exp_1 :: exp_2$	cons
	|	**struct** $\{fexp_0, \ldots, fexp_n\}$	struct
	|	$\{exp$ **with** $fexp_0, \ldots, fexp_n\}$	functional up
	|	$exp.id$	field projectio
	|	**match** $exp\{pexp_1, \ldots, pexp_n\}$	pattern match
	|	*letbind* **in** *exp*	let expression
	|	$lexp = exp$	imperative as
	|	**sizeof** *nexp*	the value of *n*
	|	**return** *exp*	return *exp* fro
	|	**exit** *exp*	halt all curren
	|	**ref** *id*	
	|	**throw** *exp*	
	|	**try** *exp* **catch** $\{pexp_1, \ldots, pexp_n\}$	
	|	**assert** (exp, exp')	halt with erro
	|	(exp)	S
	|	**var** $lexp = exp$ **in** exp'	This is an int
	|	**let** $pat = exp$ **in** exp'	This is an int
	|	**return_int** (exp)	For internal u
	|	*value*	For internal u
	|	**constraint** *n_constraint*	

exp	::=		
	|	$annot\ exp\_aux$	
$lexp\_aux$	::=		lvalue expression
	|	id	identifier
	|	**deref** exp	
	|	$id(exp_1, .., exp_n)$	memory or register write via function call
	|	$(typ)id$	
	|	$(lexp_0, .., lexp_n)$	multiple (non-memory) assignment
	|	$lexp_1 @ \ldots @ lexp_n$	vector concatenation L-exp
	|	$lexp[exp]$	vector element
	|	$lexp[exp_1..exp_2]$	subvector
	|	$lexp.id$	struct field
$lexp$	::=		
	|	$annot\ lexp\_aux$	
$fexp\_aux$	::=		field expression
	|	$id = exp$	
$fexp$	::=		
	|	$annot\ fexp\_aux$	
$opt\_default\_aux$	::=		optional default value for indexed vector expressions
	|		
	|	; **default** $= exp$	
$opt\_default$	::=		
	|	$annot\ opt\_default\_aux$	
$pexp\_aux$	::=		pattern match
	|	$pat \rightarrow exp$	
	|	pat **when** $exp_1 \rightarrow exp$	
$pexp$	::=		
	|	$annot\ pexp\_aux$	
$tannot\_opt\_aux$	::=		optional type annotation for functions
	|		
	|	$typquant\ typ$	
$tannot\_opt$	::=		
	|	$l\ tannot\_opt\_aux$	
rec_opt_aux	::=		optional recursive annotation for functions
	|		non-recursive
	|	**rec**	recursive without termination measure
	|	$\{pat \rightarrow exp\}$	recursive with termination measure

rec_opt	::=		
	|	$l\ rec\_opt\_aux$	
$effect\_opt\_aux$	::=		optional effect annotation for functions
	|		no effect annotation
	|	**effect** $effect$	
$effect\_opt$	::=		
	|	$l\ effect\_opt\_aux$	
$pexp\_funcl$	::=		
	|	$pat = exp$	
	|	$(pat$ **when** $exp_1) = exp$	
$funcl\_aux$	::=		function clause
	|	$id\ pexp\_funcl$	
$funcl$	::=		
	|	$annot\ funcl\_aux$	
$fundef\_aux$	::=		function definition
	|	**function** $rec\_opt\ tannot\_opt\ effect\_opt\ funcl_1$ **and** ... **and** $funcl_n$	
$fundef$	::=		
	|	$annot\ fundef\_aux$	
$mpat\_aux$	::=		Mapping pattern. Mostly the same as normal patterns b
	|	lit	
	|	id	
	|	$id(mpat_1, \ldots, mpat_n)$	
	|	$[mpat_1, \ldots, mpat_n]$	
	|	$mpat_1 @ \ldots @ mpat_n$	
	|	$(mpat_1, \ldots, mpat_n)$	
	|	$[\|\|mpat_1, \ldots, mpat_n\|\|]$	
	|	$(mpat)$	S
	|	$mpat_1 :: mpat_2$	
	|	$mpat_1 \uparrow\uparrow \ldots \uparrow\uparrow mpat_n$	
	|	$mpat : typ$	
	|	$mpat$ **as** id	
$mpat$	::=		
	|	$annot\ mpat\_aux$	
$mpexp\_aux$	::=		
	|	$mpat$	
	|	$mpat$ **when** exp	
$mpexp$	::=		

	$\vert$	$annot\ mpexp\_aux$		
$mapcl\_aux$	::=			mapping clause (bidirectional pattern-m
	$\vert$	$mpexp_1 \leftrightarrow mpexp_2$		
	$\vert$	$mpexp \Rightarrow exp$		
	$\vert$	$mpexp < -exp$		
$mapcl$	::=			
	$\vert$	$annot\ mapcl\_aux$		
$mapdef\_aux$	::=			mapping definition (bidirectional patter
	$\vert$	**mapping** $id\ tannot\_opt = \{mapcl_1, \ldots, mapcl_n\}$		
$mapdef$	::=			
	$\vert$	$annot\ mapdef\_aux$		
$letbind\_aux$	::=			let binding
	$\vert$	**let** $pat = exp$		let, implicit type (pat must be total)
$letbind$	::=			
	$\vert$	$annot\ letbind\_aux$		
val_spec	::=			
	$\vert$	val_spec_aux		
val_spec_aux	::=			value type specification
	$\vert$	**val** $typschm\ id$	S	specify the type of an upcoming defin
	$\vert$	**val cast** $typschm\ id$	S	
	$\vert$	**val extern** $typschm\ id$	S	specify the type of an external functi
	$\vert$	**val extern** $typschm\ id = string$	S	specify the type of a function from Le
$default\_spec\_aux$	::=			default kinding or typing assumption
	$\vert$	**default Order** $order$		
$default\_spec$	::=			
	$\vert$	$l\ default\_spec\_aux$		
$scattered\_def\_aux$	::=			scattered function and union type defini
	$\vert$	**scattered function** $rec\_opt\ tannot\_opt\ effect\_opt\ id$		scattered function definition header
	$\vert$	**function clause** $funcl$		scattered function definition clause
	$\vert$	**scattered typedef** $id =$ **const union** $typquant$		scattered union definition header
	$\vert$	**union** id **member** $type\_union$		scattered union definition member
	$\vert$	**scattered mapping** $id : tannot\_opt$		
	$\vert$	**mapping clause** $id = mapcl$		

	$\vert$	**end** *id*	scattered definition end
scattered_def	::=		
	$\vert$	*annot scattered_def_aux*	
reg_id_aux	::=		
	$\vert$	*id*	
reg_id	::=		
	$\vert$	*annot reg_id_aux*	
alias_spec_aux	::=		register alias expression forms
	$\vert$	$reg\_id.id$	
	$\vert$	$reg\_id[exp]$	
	$\vert$	$reg\_id[exp..exp']$	
	$\vert$	$reg\_id : reg\_id'$	
alias_spec	::=		
	$\vert$	*annot alias_spec_aux*	
dec_spec_aux	::=		register declarations
	$\vert$	**register** $effect\ effect'\ typ\ id$	
	$\vert$	**register configuration** $id : typ = exp$	
	$\vert$	**register alias** $id = alias\_spec$	
	$\vert$	**register alias** $typ\ id = alias\_spec$	
dec_spec	::=		
	$\vert$	*annot dec_spec_aux*	
prec	::=		
	$\vert$	**infix**	
	$\vert$	**infixl**	
	$\vert$	**infixr**	
loop_measure	::=		
	$\vert$	*loop exp*	
def	::=		top-level definition
	$\vert$	*type_def*	type definition
	$\vert$	*fundef*	function definition
	$\vert$	*mapdef*	mapping definition
	$\vert$	*letbind*	value definition
	$\vert$	*val_spec*	top-level type constraint
	$\vert$	**fix** *prec num id*	fixity declaration
	$\vert$	**overload** $id[id_1; \ldots; id_n]$	operator overload specification
	$\vert$	*default_spec*	default kind and type assumptions
	$\vert$	*scattered_def*	scattered function and type definition
	$\vert$	**termination_measure** $id\ pat = exp$	separate termination measure declaratio

	|	**termination_measure** id $loop\_measure_1, .., loop\_measure_n$		separate terminatio
	|	dec_spec		register declaration
	|	$fundef_1 .. fundef_n$		internal representat
	|	$\$string_1\ string_2\ l$		compiler directive
$defs$	::=			definition sequence
	|	$def_1 .. def_n$		
γ	::=			
	|	ϵ		
	|	$\gamma, id : typ$		
	|	$id_1 : typ_1 .. id_n : typ_n$		
	|	$\gamma_1, .., \gamma_n$		
E_aux	::=			
	|	ϵ		
	|	$E, id : typ$		
	|	E_{exp}	M	
	|	E_{pat}	M	
	|	E_{pexp}	M	
	|	E_{lexp}	M	
	|	$E, n\_constraint$	M	
	|	$E, kinded\_id_1 ... kinded\_id_n$	M	
	|	E, γ	M	
	|	E, **assert** (exp)	M	
	|	$E, typquant$	M	
E	::=			
	|	$E\_aux\ annot$		
σ	::=			
	|	ϵ		
	|	$typ\_arg/kid, \sigma$		
mut_immut	::=			
	|	**mutable**		
	|	**immutable**		
$terminals$	::=			
	|	$**$		**
	|	$\geq$		
	|	$\leq$		
	|	$\rightarrow$		
	|	$\leftrightarrow$		
	|	$\Longrightarrow$		==>
	|	$\cap$		
	|	$\uplus$		
	|	$\backslash$		
	|	$\notin$		

| $\subset$
| $\neq$
| $\emptyset$
| $<$
| $>$
| $\approx$
| $\precapprox$
| $\vdash$
| $\vdash_t$
| $\vdash_n$
| $\vdash_e$
| $\vdash_o$
| $\vdash_c$
| $'$
| $\mapsto$
| $\triangleright$
| $\leadsto$
| σ
| $\Rightarrow$
| $\_$
| **effect**
| ϵ
| **consistent_increase**
| **consistent_decrease**
| $\equiv$
| $\in$
| $\mid\!\sim$
| $\sim$

$formula$ ::=
| $judgement$
| $formula_1 \quad .. \quad formula_n$
| $id / mut\_immut : typ \in E$
| $id / register : typ \in E$
| $id / enum : typ \in E$
| $id / mut\_immut \notin E$
| $(kid, kind) \ \in \ E$
| $(id, kind) \ \in \ E$
| $\{id : typ''\} : typ' \ \in \ E$
| $id(kind_1, \ ..., kind_n) \rightarrow kind \ \in \ E$
| $\mathbf{register}\, id : typ \ \in \ E$
| $\mathbf{ret\_typ}\, typ \ \in \ E$
| $\mathbf{locals}\, id_1 : typ_1 \, .. \, id_n : typ_n \ \in \ E$
| **FIXME**
| $E \models n\_constraint$
| $kid = \mathbf{kid\_for}\, id$
| $id : (typ_1, \, .., typ_n) \rightarrow typ' \ \in \ E$
| $id : typquant\, typ \ \in \ E$
| $nexp' = \mathbf{length}\, [exp_1, \ ..., exp_n]$
| $nexp = \mathbf{length}\, lit$

	$\mid$	$\textbf{default Order}\ \mathit{order} \in E$
	$\mid$	$\textbf{distinct}\ E, \gamma$
	$\mid$	$\{\overline{\mathit{kinded\_id}_i}^{\,i}.\mathit{n\_constraint}\} \sim \mathit{typ}$
	$\mid$	$\{\overline{\mathit{kinded\_id}_i}^{\,i}, \mathit{n\_constraint}.\mathit{nexp}\} \sim \mathit{typ}$
	$\mid$	$E_1 = E_2$

well_formed	::=		
	$\mid$	$E \vdash \mathit{nexp} : \mathit{kind}$	
	$\mid$	$E \vdash \mathit{n\_constraint}$	
	$\mid$	$E \vdash \mathit{typ\_arg} : \mathit{kind}$	
	$\mid$	$E \vdash \mathit{typ}$	
	$\mid$	$E \vdash \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n$	Locals variable list is well-formed
	$\mid$	$\vdash E$	Environment is well-formed

typedefns	::=	
	$\mid$	$E \vdash \mathit{typ} \rightsquigarrow \mathit{n\_constraint}$
	$\mid$	$E \vdash \mathit{typ}_1 \lesssim \mathit{typ}_2$
	$\mid$	$E \vdash \mathit{typ}_1 \sim \mathit{typ}_2$

checker	::=	
	$\mid$	$E \vdash \mathit{lit} : \mathit{typ}$
	$\mid$	$E \ \mid\!\sim\ \mathit{pat} : \mathit{typ} \rightsquigarrow \gamma$
	$\mid$	$E \vdash \mathit{pat} : \mathit{typ} \rightsquigarrow \gamma$
	$\mid$	$E \vdash \mathit{letbind} \rightsquigarrow \gamma$
	$\mid$	$E \vdash \mathit{fexp} : \mathit{typ}$
	$\mid$	$E \ \mid\!\sim\ \mathit{fexp} : \mathit{typ}$
	$\mid$	$E \sim E'$
	$\mid$	$E \ \mid\!\sim\ \mathit{exp} : \mathit{typ}$
	$\mid$	$E \vdash \mathit{exp} : \mathit{typ}$
	$\mid$	$E \ \mid\!\sim\ \mathit{lexp} : \mathit{typ} \rightsquigarrow \gamma$
	$\mid$	$E \vdash \mathit{lexp} : \mathit{typ} \rightsquigarrow \gamma$
	$\mid$	$E \ \mid\!\sim\ \mathit{pexp} : (\mathit{typ}', \mathit{typ})$

definitions	::=	
	$\mid$	$E \vdash \mathit{type\_def}$
	$\mid$	$E \vdash \mathit{scattered\_def}$
	$\mid$	$E \vdash \mathit{funcl} : \mathit{typquant}\ \mathit{typ}$
	$\mid$	$E \vdash \mathit{fundef}$
	$\mid$	$E \vdash \mathit{def}$

judgement	::=	
	$\mid$	*well_formed*
	$\mid$	*typedefns*
	$\mid$	*checker*
	$\mid$	*definitions*

user_syntax	::=	
	$\mid$	n
	$\mid$	num

$|$ *nat*
$|$ *hex*
$|$ *bin*
$|$ *string*
$|$ *regexp*
$|$ *real*
$|$ *value*
$|$ *x*
$|$ *ix*
$|$ *l*
$|$ *annot*
$|$ *id_aux*
$|$ *id*
$|$ *kid_aux*
$|$ *kid*
$|$ *kind_aux*
$|$ *kind*
$|$ *nexp_aux*
$|$ *nexp*
$|$ *order_aux*
$|$ *order*
$|$ *base_effect_aux*
$|$ *base_effect*
$|$ *effect_aux*
$|$ *effect*
$|$ *typ_aux*
$|$ *typ*
$|$ *typ_arg_aux*
$|$ *typ_arg*
$|$ *n_constraint_aux*
$|$ *n_constraint*
$|$ *kinded_id_aux*
$|$ *kinded_id*
$|$ *quant_item_aux*
$|$ *quant_item*
$|$ *typquant_aux*
$|$ *typquant*
$|$ *typschm_aux*
$|$ *typschm*
$|$ *type_def*
$|$ *type_def_aux*
$|$ *type_union_aux*
$|$ *type_union*
$|$ *index_range_aux*
$|$ *index_range*
$|$ *lit_aux*
$|$ *lit*
$|$ $;^?$
$|$ *typ_pat_aux*
$|$ *typ_pat*

$$
\begin{array}{ll}
| & \mathit{pat\_aux} \\
| & \mathit{pat} \\
| & \mathit{loop} \\
| & \mathit{internal\_loop\_measure\_aux} \\
| & \mathit{internal\_loop\_measure} \\
| & \mathit{exp\_aux} \\
| & \mathit{exp} \\
| & \mathit{lexp\_aux} \\
| & \mathit{lexp} \\
| & \mathit{fexp\_aux} \\
| & \mathit{fexp} \\
| & \mathit{opt\_default\_aux} \\
| & \mathit{opt\_default} \\
| & \mathit{pexp\_aux} \\
| & \mathit{pexp} \\
| & \mathit{tannot\_opt\_aux} \\
| & \mathit{tannot\_opt} \\
| & \mathit{rec\_opt\_aux} \\
| & \mathit{rec\_opt} \\
| & \mathit{effect\_opt\_aux} \\
| & \mathit{effect\_opt} \\
| & \mathit{pexp\_funcl} \\
| & \mathit{funcl\_aux} \\
| & \mathit{funcl} \\
| & \mathit{fundef\_aux} \\
| & \mathit{fundef} \\
| & \mathit{mpat\_aux} \\
| & \mathit{mpat} \\
| & \mathit{mpexp\_aux} \\
| & \mathit{mpexp} \\
| & \mathit{mapcl\_aux} \\
| & \mathit{mapcl} \\
| & \mathit{mapdef\_aux} \\
| & \mathit{mapdef} \\
| & \mathit{letbind\_aux} \\
| & \mathit{letbind} \\
| & \mathit{val\_spec} \\
| & \mathit{val\_spec\_aux} \\
| & \mathit{default\_spec\_aux} \\
| & \mathit{default\_spec} \\
| & \mathit{scattered\_def\_aux} \\
| & \mathit{scattered\_def} \\
| & \mathit{reg\_id\_aux} \\
| & \mathit{reg\_id} \\
| & \mathit{alias\_spec\_aux} \\
| & \mathit{alias\_spec} \\
| & \mathit{dec\_spec\_aux} \\
| & \mathit{dec\_spec} \\
| & \mathit{prec} \\
| & \mathit{loop\_measure}
\end{array}
$$

$$
\begin{array}{ll}
| & def \\
| & defs \\
| & \gamma \\
| & E\_aux \\
| & E \\
| & \sigma \\
| & mut\_immut \\
| & terminals \\
| & formula
\end{array}
$$

$\boxed{E \vdash nexp : kind}$

$$\frac{(id, kind) \in E}{E \vdash id : kind} \quad \textsc{wfne\_id}$$

$$\frac{(kid, kind) \in E}{E \vdash kid : kind} \quad \textsc{wfne\_kid}$$

$$\frac{}{E \vdash num : \mathbf{Int}} \quad \textsc{wfne\_num}$$

$$\frac{\begin{array}{c} E \vdash nexp_1 : kind_1 \quad ... \quad E \vdash nexp_n : kind_n \\ id(kind_1, ..., kind_n) \to kind \in E \end{array}}{E \vdash id(nexp_1, ..., nexp_n) : kind} \quad \textsc{wfne\_app}$$

$$\frac{\begin{array}{c} E \vdash nexp_1 : \mathbf{Int} \\ E \vdash nexp_2 : \mathbf{Int} \end{array}}{E \vdash nexp_1 * nexp_2 : \mathbf{Int}} \quad \textsc{wfne\_times}$$

$$\frac{\begin{array}{c} E \vdash nexp_1 : \mathbf{Int} \\ E \vdash nexp_2 : \mathbf{Int} \end{array}}{E \vdash nexp_1 + nexp_2 : \mathbf{Int}} \quad \textsc{wfne\_plus}$$

$$\frac{\begin{array}{c} E \vdash nexp_1 : \mathbf{Int} \\ E \vdash nexp_2 : \mathbf{Int} \end{array}}{E \vdash nexp_1 - nexp_2 : \mathbf{Int}} \quad \textsc{wfne\_minus}$$

$$\frac{E \vdash nexp : \mathbf{Int}}{E \vdash 2\uparrow nexp : \mathbf{Int}} \quad \textsc{wfne\_exp}$$

$$\frac{E \vdash nexp : \mathbf{Int}}{E \vdash -nexp : \mathbf{Int}} \quad \textsc{wfne\_neg}$$

$\boxed{E \vdash n\_constraint}$

$$\frac{\begin{array}{c} E \vdash nexp : kind \\ E \vdash nexp' : kind \end{array}}{E \vdash nexp \equiv nexp'} \quad \textsc{wfnc\_equal}$$

$$\frac{\begin{array}{c} E \vdash nexp : kind \\ E \vdash nexp' : kind \end{array}}{E \vdash nexp \geq nexp'} \quad \textsc{wfnc\_bounded\_ge}$$

$$\frac{\begin{array}{c} E \vdash nexp : kind \\ E \vdash nexp' : kind \end{array}}{E \vdash nexp > nexp'} \quad \textsc{wfnc\_bounded\_gt}$$

$$\frac{\begin{array}{c}E \vdash nexp : kind\\ E \vdash nexp' : kind\end{array}}{E \vdash\ nexp \leq nexp'}\quad \text{WFNC\_BOUNDED\_LE}$$

$$\frac{\begin{array}{c}E \vdash nexp : kind\\ E \vdash nexp' : kind\end{array}}{E \vdash\ nexp < nexp'}\quad \text{WFNC\_BOUNDED\_LT}$$

$$\frac{\begin{array}{c}E \vdash nexp : kind\\ E \vdash nexp' : kind\end{array}}{E \vdash\ nexp != nexp'}\quad \text{WFNC\_NOT\_EQUAL}$$

$$\frac{}{E \vdash\ kid\,\mathbf{IN}\,\{num_1, \ldots, num_n\}}\quad \text{WFNC\_SET}$$

$$\frac{\begin{array}{c}E \vdash n\_constraint\\ E \vdash n\_constraint'\end{array}}{E \vdash\ n\_constraint \wedge n\_constraint'}\quad \text{WFNC\_AND}$$

$$\frac{\begin{array}{c}E \vdash n\_constraint\\ E \vdash n\_constraint'\end{array}}{E \vdash\ n\_constraint | n\_constraint'}\quad \text{WFNC\_OR}$$

$$\frac{\begin{array}{c}id(kind_0, \ldots, kind_n) \to kind\ \in\ E\\ E \vdash typ\_arg_0 : kind_0 \quad \ldots \quad E \vdash typ\_arg_n : kind_n\end{array}}{E \vdash\ id(typ\_arg_0, \ldots, typ\_arg_n)}\quad \text{WFNC\_APP}$$

$$\frac{(kid, \mathbf{Bool})\ \in\ E}{E \vdash\ kid}\quad \text{WFNC\_KID}$$

$$\frac{}{E \vdash\ \mathbf{true}}\quad \text{WFNC\_TRUE}$$

$$\frac{}{E \vdash\ \mathbf{false}}\quad \text{WFNC\_FALSE}$$

$\boxed{E \vdash typ\_arg : kind}$

$$\frac{E \vdash nexp : kind}{E \vdash\ nexp : kind}\quad \text{WFTA\_NEXP}$$

$$\frac{E \vdash typ}{E \vdash\ typ : \mathbf{Type}}\quad \text{WFTA\_TYP}$$

$$\frac{}{E \vdash\ order : \mathbf{Order}}\quad \text{WFTA\_ORDER}$$

$$\frac{E \vdash n\_constraint}{E \vdash\ n\_constraint : \mathbf{Bool}}\quad \text{WFTA\_BOOL}$$

$\boxed{E \vdash typ}$

$$\frac{(id, \mathbf{Type})\ \in\ E}{E \vdash\ id}\quad \text{WFT\_ID}$$

$$\frac{(kid, \mathbf{Type})\ \in\ E}{E \vdash\ kid}\quad \text{WFT\_VAR}$$

$$\frac{\begin{array}{c}E \vdash typ_1 \quad \ldots \quad E \vdash typ_n\\ E \vdash typ\end{array}}{E \vdash\ (typ_1, \ldots, typ_n) \to typ\ \mathbf{effect}\ effect}\quad \text{WFT\_FN}$$

$$\frac{E \vdash \mathit{typ}_1 \quad E \vdash \mathit{typ}_2}{E \vdash \ \mathit{typ}_1 \leftrightarrow \mathit{typ}_2 \ \textbf{effect}\ \mathit{effect}} \quad \text{WFT\_BIDIR}$$

$$\frac{E \vdash \mathit{typ}_1 \quad .... \quad E \vdash \mathit{typ}_n}{E \vdash \ (\mathit{typ}_1,\ ....\ ,\mathit{typ}_n)} \quad \text{WFT\_TUP}$$

$$\frac{\mathit{id}(\mathit{kind}_1,\ ...\ ,\mathit{kind}_n) \to \mathit{kind} \ \in \ E \quad E \vdash \mathit{typ\_arg}_1 : \mathit{kind}_1 \quad ... \quad E \vdash \mathit{typ\_arg}_n : \mathit{kind}_n}{E \vdash \ \mathit{id}(\mathit{typ\_arg}_1,\ ...\ ,\mathit{typ\_arg}_n)} \quad \text{WFT\_APP}$$

$$\frac{E, \mathit{kinded\_id}_1 \ldots \mathit{kinded\_id}_n \vdash \mathit{n\_constraint} \quad E, \mathit{kinded\_id}_1 \ldots \mathit{kinded\_id}_n \vdash \mathit{typ}}{E \vdash \ \{\mathit{kinded\_id}_1 \ldots \mathit{kinded\_id}_n, \mathit{n\_constraint}.\mathit{typ}\}} \quad \text{WFT\_EXIST}$$

$\boxed{E \vdash \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n}$ Locals variable list is well-formed

$$\frac{}{E \vdash} \quad \text{WFLOC\_EMPTY}$$

$$\frac{E \vdash \mathit{typ} \quad E \vdash \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n}{E \vdash \mathit{id} : \mathit{typ}\ \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n} \quad \text{WFLOC\_CONS}$$

$\boxed{\vdash E}$ Environment is well-formed

$$\frac{\textbf{locals}\ \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n \ \in \ E \quad E \vdash \mathit{id}_1 : \mathit{typ}_1 \,..\, \mathit{id}_n : \mathit{typ}_n}{\vdash E} \quad \text{WFE\_WF}$$

$\boxed{E \vdash \mathit{typ} \leadsto \mathit{n\_constraint}}$

$\boxed{E \vdash \mathit{typ}_1 \lesssim \mathit{typ}_2}$

$$\frac{E \vdash \mathit{typ}_1 \leadsto \mathit{n\_constraint}_1 \quad E \vdash \mathit{typ}_2 \leadsto \mathit{n\_constraint}_2 \quad E, \mathit{n\_constraint}_1 \models \mathit{n\_constraint}_2}{E \vdash \mathit{typ}_1 \lesssim \mathit{typ}_2} \quad \text{ST\_SUBTYPE}$$

$\boxed{E \vdash \mathit{typ}_1 \sim \mathit{typ}_2}$

$\boxed{E \vdash \mathit{lit} : \mathit{typ}}$

$$\frac{}{E \vdash \ () : \ \textbf{unit}} \quad \text{CL\_UNIT}$$

$$\frac{}{E \vdash \ \textbf{bitzero} : \ \textbf{bit}} \quad \text{CL\_ZERO}$$

$$\frac{}{E \vdash \ \textbf{bitone} : \ \textbf{bit}} \quad \text{CL\_ONE}$$

$$\frac{}{E \vdash \ \textbf{true} : \ \textbf{bool}(\ \textbf{true})} \quad \text{CL\_TRUE}$$

$$\frac{}{E \vdash \ \textbf{false} : \ \textbf{bool}(\ \textbf{false})} \quad \text{CL\_FALSE}$$

$$\frac{}{E \vdash \ \mathit{num} : \ \textbf{atom}(\ \mathit{num})} \quad \text{CL\_NUM}$$

$$\frac{\textbf{default Order}\ \mathit{order} \ \in \ E \quad \mathit{nexp} = \textbf{length}\ \mathit{hex}}{E \vdash \ \mathit{hex} : \ \textbf{vector}(\mathit{nexp},\ \mathit{order},\ \textbf{bit})} \quad \text{CL\_HEX}$$

$$\frac{\begin{array}{l}\textbf{default Order}\ \mathit{order}\ \in\ E\\ \mathit{nexp} = \textbf{length}\ \mathit{bin}\end{array}}{E \vdash\ \mathit{bin} :\ \textbf{vector}(\mathit{nexp},\ \mathit{order},\ \textbf{bit})}\quad\text{CL\_BIN}$$

$$\frac{}{E \vdash\ \mathit{string}_1 :\ \mathit{string}}\quad\text{CL\_STRING}$$

$$\frac{}{E \vdash\ \mathit{real} :\ \mathit{real}}\quad\text{CL\_REAL}$$

$\boxed{E\ \mid\!\sim\ \mathit{pat} : \mathit{typ} \leadsto \gamma}$

$$\frac{\begin{array}{l}E_{\mathit{pat}}\ \vdash \mathit{pat} :\ \mathrm{typ}_{\mathit{pat}} \leadsto \gamma\\ E_{\mathit{pat}}\ \vdash \mathit{typ}' \lesssim \mathit{typ}\\ E \sim E_{\mathit{pat}}\end{array}}{E\ \mid\!\sim\ \mathit{pat} : \mathit{typ} \leadsto \gamma}\quad\text{CPS\_CHECK\_PAT}$$

$\boxed{E \vdash \mathit{pat} : \mathit{typ} \leadsto \gamma}$

$$\frac{E \vdash \mathit{lit} : \mathit{typ}}{E \vdash\ \mathit{lit} : \mathit{typ} \leadsto \epsilon}\quad\text{CP\_LIT}$$

$$\frac{}{E \vdash\ \_ : \mathit{typ} \leadsto \epsilon}\quad\text{CP\_WILD}$$

$$\frac{\begin{array}{c}E\ \mid\!\sim\ \mathit{pat}_1 : \mathit{typ} \leadsto \gamma\\ E\ \mid\!\sim\ \mathit{pat}_2 : \mathit{typ} \leadsto \gamma\end{array}}{E \vdash\ \mathit{pat}_1 | \mathit{pat}_2 : \mathit{typ} \leadsto \gamma}\quad\text{CP\_OR}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat} : \mathit{typ} \leadsto \gamma}{E \vdash\!\sim \mathit{pat} : \mathit{typ} \leadsto \epsilon}\quad\text{CP\_NOT}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat} : \mathit{typ} \leadsto \gamma}{E \vdash\ (\mathit{pat}\ \textbf{as}\ \mathit{id}) : \mathit{typ} \leadsto \gamma, \mathit{id} : \mathit{typ}}\quad\text{CP\_AS}$$

$$\frac{}{E \vdash\ \mathit{id} : \mathit{typ} \leadsto \mathit{id} : \mathit{typ}}\quad\text{CP\_ID}$$

$$\frac{\begin{array}{l}\mathit{id} : (\mathit{typ}_1, .., \mathit{typ}_n) \rightarrow \mathit{typ}\ \in\ E\\ E\ \mid\!\sim\ \mathit{pat}_1 : \mathit{typ}_1 \leadsto \gamma_1\quad ..\quad E\ \mid\!\sim\ \mathit{pat}_n : \mathit{typ}_n \leadsto \gamma_n\end{array}}{E \vdash\ \mathit{id}(\mathit{pat}_1, .., \mathit{pat}_n) : \mathit{typ} \leadsto \gamma_1, .., \gamma_n}\quad\text{CP\_APP}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat}_1 : \mathit{typ} \leadsto \gamma_1\quad ...\quad E\ \mid\!\sim\ \mathit{pat}_n : \mathit{typ} \leadsto \gamma_n}{E \vdash\ [\mathit{pat}_1, ..., \mathit{pat}_n] :\ \textbf{vector}(\mathit{nexp}_n,\ \mathit{order},\ \mathit{typ}) \leadsto \gamma_1, ..., \gamma_n}\quad\text{CP\_VECTOR}$$

$$\frac{\begin{array}{l}\overline{E\ \mid\!\sim\ \mathit{pat}_i :\ \textbf{vector}(\mathit{nexp}_i,\ \mathit{order},\ \mathit{typ}) \leadsto \gamma_i}^{\,i \in 1...n}\\ E \models\ \mathit{nexp}_1 + ... + \mathit{nexp}_n \equiv \mathit{nexp}\end{array}}{E \vdash\ \mathit{pat}_1 @ ... @ \mathit{pat}_n :\ \textbf{vector}(\mathit{nexp},\ \mathit{order},\ \mathit{typ}) \leadsto \gamma_1, ..., \gamma_n}\quad\text{CP\_VECTOR\_CONCAT}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat}_1 : \mathit{typ}_1 \leadsto \gamma_1\quad ..\quad E\ \mid\!\sim\ \mathit{pat}_n : \mathit{typ}_n \leadsto \gamma_n}{E \vdash\ (\mathit{pat}_1, .., \mathit{pat}_n) :\ (\mathit{typ}_1, .., \mathit{typ}_n) \leadsto \gamma_1, .., \gamma_n}\quad\text{CP\_TUP}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat}_1 : \mathit{typ} \leadsto \gamma_1\quad ...\quad E\ \mid\!\sim\ \mathit{pat}_n : \mathit{typ} \leadsto \gamma_n}{E \vdash\ [||\mathit{pat}_1, ..., \mathit{pat}_n||] :\ \textbf{list}(\mathit{typ}) \leadsto \gamma_1, ..., \gamma_n}\quad\text{CP\_LIST}$$

$$\frac{E\ \mid\!\sim\ \mathit{pat}_1 :\ \mathit{string} \leadsto \gamma_1\quad ...\quad E\ \mid\!\sim\ \mathit{pat}_n :\ \mathit{string} \leadsto \gamma_n}{E \vdash\ \mathit{pat}_1 \uparrow\uparrow ... \uparrow\uparrow \mathit{pat}_n :\ \mathit{string} \leadsto \gamma_1, ..., \gamma_n}\quad\text{CP\_STRING\_APPEND}$$

$\boxed{E \vdash \mathit{letbind} \leadsto \gamma}$

$$\frac{\begin{array}{l} E \mathrel{|\!\!\sim} pat : \mathrm{typ}_{exp} \rightsquigarrow \gamma \\ E \mathrel{|\!\!\sim} exp : \mathrm{typ}_{exp} \end{array}}{E \vdash \mathbf{let}\, pat = exp \rightsquigarrow \gamma} \quad \text{CL\_VAL}$$

$\boxed{E \vdash fexp : typ}$

$$\frac{\begin{array}{l} \{id : typ'\} : typ \in E \\ E \mathrel{|\!\!\sim} exp : typ' \end{array}}{E \vdash id = exp : typ} \quad \text{CFE\_FEXP}$$

$\boxed{E \mathrel{|\!\!\sim} fexp : typ}$

$$\frac{\begin{array}{l} E' \vdash fexp : typ' \\ E' \vdash typ' \lesssim typ \\ E \sim E' \end{array}}{E \mathrel{|\!\!\sim} fexp : typ} \quad \text{CFS\_FEXP}$$

$\boxed{E \sim E'}$

$\boxed{E \mathrel{|\!\!\sim} exp : typ}$

$$\frac{\begin{array}{l} E_{exp} \vdash exp : \mathrm{typ}_{exp} \\ E_{exp} \vdash \mathrm{typ}_{exp} \lesssim typ \\ E \sim E_{exp} \end{array}}{E \mathrel{|\!\!\sim} exp : typ} \quad \text{CES\_CHECK\_EXP}$$

$\boxed{E \vdash exp : typ}$

$$\frac{E \mathrel{|\!\!\sim} exp : typ}{E \vdash \{exp\} : typ} \quad \text{CE\_BLOCK\_SINGLE}$$

$$\frac{\begin{array}{l} E \mathrel{|\!\!\sim} exp : \mathbf{unit} \\ E \mathrel{|\!\!\sim} \{exp_1; \ldots; exp_n\} : typ \end{array}}{E \vdash \{exp; exp_1; \ldots; exp_n\} : typ} \quad \text{CE\_BLOCK\_CONS}$$

$$\frac{id / mut\_immut : typ \in E}{E \vdash id : typ} \quad \text{CE\_ID}$$

$$\frac{id / enum : typ \in E}{E \vdash id : typ} \quad \text{CE\_ENUM}$$

$$\frac{id / register : typ \in E}{E \vdash id : typ} \quad \text{CE\_REGISTER}$$

$$\frac{id / register : typ \in E}{E \vdash \mathbf{ref}\, id : \mathbf{register}(typ)} \quad \text{CE\_REF}$$

$$\frac{E \vdash lit : typ}{E \vdash lit : typ} \quad \text{CE\_LIT}$$

$$\frac{\begin{array}{l} E \mathrel{|\!\!\sim} exp : typ' \\ E \vdash typ' \lesssim typ \end{array}}{E \vdash (typ')exp : typ} \quad \text{CE\_CAST}$$

$$\frac{\begin{array}{l} id : (typ_1, .., typ_n) \to typ' \in E \\ E \mathrel{|\!\!\sim} exp_1 : \sigma(typ_1) \quad .. \quad E \mathrel{|\!\!\sim} exp_n : \sigma(typ_n) \\ E \vdash \sigma(typ') \lesssim typ \end{array}}{E \vdash id(exp_1, .., exp_n) : typ} \quad \text{CE\_APP}$$

$$\frac{E \;|\!\sim\; exp_1 : typ_1 \quad .... \quad E \;|\!\sim\; exp_n : typ_n}{E \vdash (exp_1, ..., exp_n) : (typ_1, ..., typ_n)} \quad \text{CE\_TUPLE}$$

$$\frac{\begin{array}{l} \{kinded\_id_1 .. kinded\_id_n.n\_constraint\} \sim \mathrm{typ}_{exp_1} \\ E \;|\!\sim\; exp_1 : \mathrm{typ}_{exp_1} \\ E, n\_constraint \;|\!\sim\; exp_2 : typ \\ E, \; \mathbf{not}(\, n\_constraint) \;|\!\sim\; exp_3 : typ \end{array}}{E \vdash \mathbf{if}\; exp_1 \;\mathbf{then}\; exp_2 \;\mathbf{else}\; exp_3 : typ} \quad \text{CE\_IF}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp_1 : \;\mathbf{bool} \\ E, \mathbf{assert}\,(exp_1) \vdash exp_2 : \;\mathbf{unit} \end{array}}{E \vdash loop \;\; exp_1 \; exp_2 : \;\mathbf{unit}} \quad \text{CE\_LOOP\_NO\_MEASURE}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp : \;\mathbf{int} \\ E \;|\!\sim\; exp_1 : \;\mathbf{bool} \\ E, \mathbf{assert}\,(exp_1) \vdash exp_2 : \;\mathbf{unit} \end{array}}{E \vdash loop \;\mathbf{termination\_measure}\,\{exp\}\; exp_1 \; exp_2 : \;\mathbf{unit}} \quad \text{CE\_LOOP\_MEASURE}$$

$$\frac{\begin{array}{l} \{\,\overline{kinded\_id_i}^{\,i}, n\_constraint_1.nexp_1\} \sim \mathrm{typ}_{exp_1} \\ E \;|\!\sim\; exp_1 : \mathrm{typ}_{exp_1} \\ \{\,\overline{kinded\_id'_i}^{\,i}, n\_constraint_2.nexp_2\} \sim \mathrm{typ}_{exp_2} \\ E \;|\!\sim\; exp_2 : \mathrm{typ}_{exp_2} \\ E \;|\!\sim\; exp_3 : \;\mathbf{int} \\ E' = E, \overline{kinded\_id_i}^{\,i}, \overline{kinded\_id'_i}^{\,i}, n\_constraint_1 \wedge n\_constraint_2 \,, id : \;\mathbf{range}(\, nexp_1, \, nexp_2) \\ E' \;|\!\sim\; exp_4 : \;\mathbf{unit} \end{array}}{E \vdash \mathbf{foreach}\,(id \;\mathbf{from}\; exp_1 \;\mathbf{to}\; exp_2 \;\mathbf{by}\; exp_3 \;\mathbf{in}\; order)exp_4 : \;\mathbf{unit}} \quad \text{CE\_FOR}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp_1 : typ \quad ... \quad E \;|\!\sim\; exp_n : typ \\ nexp' = \;\mathbf{length}\,[exp_1, ..., exp_n] \\ E \models \; nexp \equiv nexp' \end{array}}{E \vdash [exp_1, ..., exp_n] : \;\mathbf{vector}(\, nexp, \, order, \, typ)} \quad \text{CE\_VECTOR}$$

$$\frac{E \;|\!\sim\; exp_1 : typ \quad .. \quad E \;|\!\sim\; exp_n : typ}{E \vdash [|exp_1, .., exp_n|] : \;\mathbf{list}(\, typ)} \quad \text{CE\_LIST}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp_1 : typ \\ E \;|\!\sim\; exp_2 : \;\mathbf{list}(\, typ) \end{array}}{E \vdash \; exp_1 :: exp_2 : \;\mathbf{list}(\, typ)} \quad \text{CE\_CONS}$$

$$\frac{E \;|\!\sim\; fexp_0 : typ \quad ... \quad E \;|\!\sim\; fexp_n : typ}{E \vdash \mathbf{struct}\,\{fexp_0, ..., fexp_n\} : typ} \quad \text{CE\_RECORD}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp : typ \\ E \;|\!\sim\; fexp_0 : typ \quad ... \quad E \;|\!\sim\; fexp_n : typ \end{array}}{E \vdash \{exp \;\mathbf{with}\; fexp_0, ..., fexp_n\} : typ} \quad \text{CE\_RECORD\_UPDATE}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp : typ' \\ \{id : typ\} : typ' \in E \end{array}}{E \vdash \; exp.id : typ} \quad \text{CE\_FIELD}$$

$$\frac{\begin{array}{l} E \;|\!\sim\; exp : typ' \\ E \;|\!\sim\; pexp_1 : (typ', typ) \quad ... \quad E \;|\!\sim\; pexp_n : (typ', typ) \end{array}}{E \vdash \mathbf{match}\; exp\{pexp_1, ..., pexp_n\} : typ} \quad \text{CE\_CASE}$$

$$\frac{\begin{array}{l} E \vdash \mathit{letbind} \leadsto \gamma \\ \textbf{distinct}\, E, \gamma \\ E, \gamma \ \mid\!\sim\ \mathit{exp} : \mathit{typ} \end{array}}{E \vdash\ \mathit{letbind}\,\textbf{in}\,\mathit{exp} : \mathit{typ}} \quad \text{CE\_LET}$$

$$\frac{\begin{array}{l} E\ \mid\!\sim\ \mathit{exp}_1 : \mathrm{typ}_{\mathit{exp}_1} \\ E\ \mid\!\sim\ \mathit{lexp} : \mathrm{typ}_{\mathit{exp}_1} \leadsto \gamma \\ \textbf{distinct}\, E, \gamma \\ E, \gamma\ \mid\!\sim\ \mathit{exp}_2 : \mathit{typ} \end{array}}{E \vdash\ \textbf{var}\,\mathit{lexp} = \mathit{exp}_1\,\textbf{in}\,\mathit{exp}_2 : \mathit{typ}} \quad \text{CE\_ASSIGN}$$

$$\frac{\begin{array}{l} \textbf{ret\_typ}\,\mathit{typ}' \in E \\ E\ \mid\!\sim\ \mathit{exp} : \mathit{typ}' \end{array}}{E \vdash\ \textbf{return}\,\mathit{exp} : \mathit{typ}} \quad \text{CE\_RETURN}$$

$$\frac{E\ \mid\!\sim\ \mathit{exp} :\ \textbf{unit}}{E \vdash\ \textbf{exit}\,\mathit{exp} : \mathit{typ}} \quad \text{CE\_EXIT}$$

$$\frac{E\ \mid\!\sim\ \mathit{exp} :\ \textbf{exception}}{E \vdash\ \textbf{throw}\,\mathit{exp} : \mathit{typ}} \quad \text{CE\_THROW}$$

$$\frac{\begin{array}{l} E\ \mid\!\sim\ \mathit{exp} : \mathit{typ} \\ E\ \mid\!\sim\ \mathit{pexp}_1 : (\ \textbf{exception}, \mathit{typ}) \quad \ldots \quad E\ \mid\!\sim\ \mathit{pexp}_n : (\ \textbf{exception}, \mathit{typ}) \end{array}}{E \vdash\ \textbf{try}\,\mathit{exp}\,\textbf{catch}\,\{\mathit{pexp}_1, \ldots, \mathit{pexp}_n\} : \mathit{typ}} \quad \text{CE\_TRY}$$

$$\frac{\begin{array}{l} E\ \mid\!\sim\ \mathit{exp} :\ \textbf{bool}(\ \mathit{n\_constraint}) \\ E\ \mid\!\sim\ \mathit{exp}' :\ \mathit{string} \\ E \models \mathit{n\_constraint} \end{array}}{E \vdash\ \textbf{assert}\,(\mathit{exp}, \mathit{exp}') :\ \textbf{unit}} \quad \text{CE\_ASSERT}$$

$$\frac{\begin{array}{l} E\ \mid\!\sim\ \mathit{exp}' : \mathit{typ}' \\ E \vdash \mathit{lexp} : \mathit{typ}' \leadsto \gamma \\ E, \gamma\ \mid\!\sim\ \mathit{exp} : \mathit{typ} \end{array}}{E \vdash\ \textbf{var}\,\mathit{lexp} = \mathit{exp}'\,\textbf{in}\,\mathit{exp} : \mathit{typ}} \quad \text{CE\_VAR}$$

$\boxed{E\ \mid\!\sim\ \mathit{lexp} : \mathit{typ} \leadsto \gamma}$

$$\frac{\begin{array}{l} E_{\mathit{lexp}} \vdash \mathit{lexp} :\ \mathrm{typ}_{\mathit{lexp}} \leadsto \gamma \\ E \vdash\ \mathrm{typ}_{\mathit{lexp}} \lesssim \mathit{typ} \\ E \sim E_{\mathit{lexp}} \end{array}}{E\ \mid\!\sim\ \mathit{lexp} : \mathit{typ} \leadsto \gamma} \quad \text{CLES\_CHECK\_EXP}$$

$\boxed{E \vdash \mathit{lexp} : \mathit{typ} \leadsto \gamma}$

$$\frac{\mathit{id}/\textbf{mutable} \notin E}{E \vdash\ \mathit{id} : \mathit{typ} \leadsto \mathit{id} : \mathit{typ}} \quad \text{CLE\_ID\_NB}$$

$$\frac{\begin{array}{l} \mathit{id}/\textbf{mutable} : \mathit{typ}' \in E \\ E \vdash \mathit{typ} \lesssim \mathit{typ}' \end{array}}{E \vdash\ \mathit{id} : \mathit{typ} \leadsto \epsilon} \quad \text{CLE\_ID\_B}$$

$$\frac{\begin{array}{l} \mathit{id}/\textbf{mutable} \notin E \\ E \vdash \mathit{typ}' \lesssim \mathit{typ} \end{array}}{E \vdash\ (\mathit{typ})\mathit{id} : \mathit{typ}' \leadsto \mathit{id} : \mathit{typ}} \quad \text{CLE\_CAST\_NB}$$

$$\frac{\begin{array}{l} id/\mathbf{mutable} : typ'' \in E \\ E \vdash typ \lesssim typ'' \\ E \vdash typ' \lesssim typ \end{array}}{E \vdash \ (typ)id : typ' \leadsto \epsilon} \quad \text{CLE\_CAST\_B}$$

$$\frac{id/register : typ \in E}{E \vdash \ id : typ \leadsto \epsilon} \quad \text{CLE\_REG}$$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ exp : \ \mathbf{register}(\ typ') \\ E \vdash typ' \lesssim typ \end{array}}{E \vdash \mathbf{deref}\ exp : typ \leadsto \epsilon} \quad \text{CLE\_DEREF}$$

$$\frac{E \ \mid\!\sim \ lexp_0 : typ_0 \leadsto \gamma_0 \quad .. \quad E \ \mid\!\sim \ lexp_n : typ_n \leadsto \gamma_n}{E \vdash \ (lexp_0, \, .., lexp_n) : \ (typ_0, \, .., typ_n) \leadsto \gamma_0, \, .., \gamma_n} \quad \text{CLE\_TUP}$$

$$\frac{\begin{array}{l} \overline{E \ \mid\!\sim \ lexp_i : \ \mathbf{vector}(\ nexp_i, \ order, \ typ) \leadsto \gamma_i}^{i \in 1 \ldots n} \\ E \models \ nexp \equiv \ nexp_1 + \ \ldots \ + nexp_n \end{array}}{E \vdash \ lexp_1 @ \ldots @ lexp_n : \ \mathbf{vector}(\ nexp, \ order, \ typ) \leadsto \gamma_1, \, \ldots, \gamma_n} \quad \text{CLE\_VECTOR\_CONCAT}$$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ lexp : \ \mathbf{vector}(\ nexp, \ order, \ typ) \leadsto \gamma \\ E \ \mid\!\sim \ exp : \ \mathbf{int} \end{array}}{E \vdash \ lexp[exp] : typ \leadsto \gamma} \quad \text{CLE\_VECTOR}$$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ lexp : \ \mathbf{vector}(\ nexp', \ order, \ typ) \leadsto \gamma \\ E \ \mid\!\sim \ exp_1 : \ \mathbf{int} \\ E \ \mid\!\sim \ exp_2 : \ \mathbf{int} \end{array}}{E \vdash \ lexp[exp_1..exp_2] : \ \mathbf{vector}(\ nexp, \ order, \ typ) \leadsto \gamma} \quad \text{CLE\_VECTOR\_RANGE}$$

$$\frac{}{E \vdash \ lexp.id : typ \leadsto \gamma} \quad \text{CLE\_FIELD}$$

$\boxed{E \ \mid\!\sim \ pexp : (typ', typ)}$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ pat : typ' \leadsto \gamma \\ \mathbf{distinct}\ E, \gamma \\ E, \gamma \ \mid\!\sim \ exp : typ \end{array}}{E \ \mid\!\sim \ pat \rightarrow exp : (typ', typ)} \quad \text{CPE\_EXP}$$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ pat : typ' \leadsto \gamma \\ \mathbf{distinct}\ E, \gamma \\ E, \gamma \ \mid\!\sim \ exp_1 : \ \mathbf{bool} \\ E, \gamma \ \mid\!\sim \ exp_2 : typ \end{array}}{E \ \mid\!\sim \ pat\, \mathbf{when}\, exp_1 \rightarrow exp_2 : (typ', typ)} \quad \text{CPE\_WHEN}$$

$\boxed{E \vdash type\_def}$

$\boxed{E \vdash scattered\_def}$

$\boxed{E \vdash funcl : typquant\ typ}$

$$\frac{\begin{array}{l} E, typquant \ \mid\!\sim \ pat : \ (typ_1, \, \ldots, typ_n) \leadsto \gamma \\ E, typquant\,, \gamma \ \mid\!\sim \ exp : typ \end{array}}{E \vdash \ id\ pat = exp : typquant\ (typ_1, \, \ldots, typ_n) \rightarrow typ\, \mathbf{effect}\ effect} \quad \text{CFCL\_EXP}$$

$$\frac{\begin{array}{l} E \ \mid\!\sim \ exp_1 : \ \mathbf{bool} \\ E, typquant \ \mid\!\sim \ pat : \ (typ_1, \, \ldots, typ_n) \leadsto \gamma \\ E, typquant\,, \gamma \vdash typ \end{array}}{E \vdash \ id\ (pat\, \mathbf{when}\, exp_1) = exp : typquant\ (typ_1, \, \ldots, typ_n) \rightarrow typ\, \mathbf{effect}\ effect} \quad \text{CFCL\_WHEN}$$

$\boxed{E \vdash \mathit{fundef}}$

$$\frac{\begin{array}{l} id : typquant\ typ \in E \\ E \vdash id\ pexp\_funcl_1 : typquant\ typ \quad \ldots \quad E \vdash id\ pexp\_funcl_n : typquant\ typ \end{array}}{E \vdash \textbf{function}\ rec\_opt\ \ effect\_opt\ \ id\ pexp\_funcl_1\ \textbf{and} \ldots \textbf{and}\ id\ pexp\_funcl_n} \quad \textsc{CFD\_NONE}$$

$$\frac{E \vdash funcl_1 : typquant\ typ \quad \ldots \quad E \vdash funcl_n : typquant\ typ}{E \vdash \textbf{function}\ rec\_opt\ \ typquant\ typ\ effect\_opt\ funcl_1\ \textbf{and} \ldots \textbf{and}\ funcl_n} \quad \textsc{CFD\_TANNOT}$$

$\boxed{E \vdash \mathit{def}}$

$$\frac{}{E \vdash type\_def} \quad \textsc{CD\_TYPE}$$

$$\frac{E \vdash fundef}{E \vdash fundef} \quad \textsc{CD\_FUNDEF}$$

$$\frac{}{E \vdash letbind} \quad \textsc{CD\_VAL}$$

$$\frac{}{E \vdash val\_spec} \quad \textsc{CD\_SPEC}$$

$$\frac{}{E \vdash default\_spec} \quad \textsc{CD\_DEFAULT}$$

$$\frac{}{E \vdash scattered\_def} \quad \textsc{CD\_SCATTERED}$$

$$\frac{}{E \vdash dec\_spec} \quad \textsc{CD\_REG\_DEF}$$

```
Definition rules:        118 good    0 bad
Definition rule clauses: 300 good    0 bad
```