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
q	Index variables for meta-lists
$\dot{ctor}$	Constructor
x, y, z, x, f, ka	Identifier
bit	Bit
u	Mutable Variables
β	Base type variables
tid	Type ID

l	::=			source location
	|			
annot	::=			
	|			
kid	::=			kinded IDs: Type, Int, and Order variables
	|	$'x$		
kind	::=			base kind
	|	**Type**		kind of types
	|	**Int**		kind of natural number size expressions
	|	**Order**		kind of vector order specifications
	|	**Bool**		kind of constraints
nexp	::=			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
	|	$nexp_1 - nexp_2$		subtraction
	|	$2\uparrow nexp$		exponential
	|	$-nexp$		unary negation
	|	$(nexp)$	S	
	|	$nexp_1 + \ldots + nexp_n$	M	
order	::=			vector order specifications, of kind Order
	|	*kid*		variable
	|	**inc**		increasing
	|	**dec**		decreasing
	|	$(order)$	S	
base_effect	::=			effect
	|	**rreg**		read register
	|	**wreg**		write register
	|	**rmem**		read memory
	|	**rmemt**		read memory and tag
	|	**wmem**		write memory
	|	**wmea**		signal effective address for writing memory
	|	**exmem**		determine if a store-exclusive (ARM) is going to succeed
	|	**wmv**		write memory, sending only value
	|	**wmvt**		write memory, sending only value and tag
	|	**barr**		memory barrier
	|	**depend**		dynamic footprint
	|	**undef**		undefined-instruction exception
	|	**unspec**		unspecified values
	|	**nondet**		nondeterminism, from **nondet**

	| **escape**		potential exception
	| **config**		configuration option
effect	::=		
	| $\{base\_effect_1, .., base\_effect_n\}$		effect set
	| **pure**	M	sugar for empty effect set
typ	::=		type expressions, of kind Type
	|		
	| *id*		defined type
	| *kid*		type variable
	| $(typ_1, ..., typ_n) \rightarrow typ_2$ **effect** *effect*		Function (first-order only)
	| $typ_1 \leftrightarrow typ_2$ **effect** *effect*		Mapping
	| $(typ_1, ..., typ_n)$		Tuple
	| $id(typ\_arg_1, ..., typ\_arg_n)$		type constructor application
	| (typ)	S	
	| $\{kinded\_id_1 ... kinded\_id_n, n\_constraint.typ\}$		
	| typ_{exp}	M	
	| typ_{lexp}	M	
	| typ_{pat}	M	
	| **sigma** (typ)	M	
typ_arg	::=		type constructor arguments of
	| *nexp*		
	| *typ*		
	| *order*		
	| *n_constraint*		
n_constraint, *ncons*	::=		constraint over kind Int
	| $nexp \equiv nexp'$		
	| $nexp \geq nexp'$		
	| $nexp > nexp'$		
	| $nexp \leq nexp'$		
	| $nexp < nexp'$		
	| $nexp! = nexp'$		
	| kid **IN** $\{num_1, ..., num_n\}$		
	| $n\_constraint \wedge n\_constraint'$		
	| $n\_constraint \| n\_constraint'$		
	| $id(typ\_arg_0, ..., typ\_arg_n)$		
	| *kid*		
	| **true**		
	| **false**		
kinded_id	::=		optionally kind-annotated ide
	| *kind kid*		kind-annotated variable
	| *kid*	S	
quant_item	::=		kinded identifier or Int constr
	| *kinded_id*		optionally kinded identifier

	$n\_constraint$	constraint
|	$kinded\_id_0 \ldots kinded\_id_n$	
typquant	::=	type quantifiers and constraints
|	$\forall\ quant\_item_1, \ldots, quant\_item_n.$	
|		empty
typschm	::=	type scheme
|	*typquant typ*	
type_def	::=	
|	*type_def_aux*	
type_def_aux	::=	type definition body
|	**type** *id typquant* = *typ_arg*	
		type abbreviation
|	**typedef** *id* = **const struct** *typquant*{$typ_1\ id_1; \ldots; typ_n\ id_n$;?}	
		struct type definition
|	**typedef** *id* = **const union** *typquant*{$type\_union_1; \ldots; type\_union_n$;?}	
		tagged union type definition
|	**typedef** *id* = **enumerate** {$id_1; \ldots; id_n$;?}	
		enumeration type definition
|	**bitfield** $id : typ = \{id_1 : index\_range_1, \ldots, id_n : index\_range_n\}$	
		register mutable bitfield type definition
type_union	::=	type union constructors
|	*typ id*	
index_range	::=	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
lit	::=	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*	
;?	::=	optional semi-colon
|		
|	;	

typ_pat	::=		type pattern
	|	_	
	|	*kid*	
	|	$id(typ\_pat_1, .., typ\_pat_n)$	
pat	::=		pattern
	|	*lit*	literal constant pa
	|	_	wildcard
	|	$pat_1 \mid pat_2$	pattern disjunctio
	|	$\sim pat$	pattern negation
	|	$(pat\ \mathbf{as}\ id)$	named pattern
	|	$(typ)pat$	typed pattern
	|	*id*	identifier
	|	$pat\ typ\_pat$	bind pattern to ty
	|	$id(pat_1, .., pat_n)$	union constructor
	|	$[pat_1, ..., pat_n]$	vector pattern
	|	$pat_1 @ ... @ pat_n$	concatenated vect
	|	$(pat_1, ..., pat_n)$	tuple pattern
	|	$[\mid\mid pat_1, .., pat_n \mid\mid]$	list pattern
	|	(pat)	S
	|	$pat_1 :: pat_2$	Cons patterns
	|	$pat_1 \uparrow\uparrow ... \uparrow\uparrow pat_n$	string append pat
loop	::=		
	|	**while**	
	|	**until**	
internal_loop_measure	::=		internal syntax for a
	|		
	|	**termination_measure** $\{exp\}$	
exp	::=		expression
	|	$\{exp_1; ...; exp_n\}$	sequential block
	|	*id*	identifier
	|	*lit*	literal constant
	|	$(typ)exp$	cast
	|	$id(exp_1, .., exp_n)$	function applicatio
	|	$exp_1\ id\ exp_2$	infix function appl
	|	$(exp_1, ..., exp_n)$	tuple
	|	$\mathbf{if}\ exp_1\ \mathbf{then}\ exp_2\ \mathbf{else}\ exp_3$	conditional
	|	$loop\ internal\_loop\_measure\ exp_1\ exp_2$	
	|	$\mathbf{foreach}\ (id\ \mathbf{from}\ exp_1\ \mathbf{to}\ exp_2\ \mathbf{by}\ exp_3\ \mathbf{in}\ order)exp_4$	for loop
	|	$[exp_1, ..., exp_n]$	vector (indexed fr
	|	$exp[exp']$	vector access
	|	$exp[exp_1..exp_2]$	subvector extracti
	|	$[exp\ \mathbf{with}\ exp_1 = exp_2]$	vector functional u
	|	$[exp\ \mathbf{with}\ exp_1..exp_2 = exp_3]$	vector subrange u
	|	$exp_1 @ exp_2$	vector concatenati
	|	$[\mid exp_1, .., exp_n \mid]$	list
	|	$exp_1 :: exp_2$	cons

	struct $\{fexp_0, \ldots, fexp_n\}$	struct
	$\{exp$ **with** $fexp_0, \ldots, fexp_n\}$	functional update of struct
	$exp.id$	field projection from struct
	match $exp\{pexp_1, \ldots, pexp_n\}$	pattern matching
	$letbind$ **in** exp	let expression
	$lexp = exp$	imperative assignment
	sizeof $nexp$	the value of $nexp$ at run time
	return exp	return exp from current function
	exit exp	halt all current execution
	ref id	
	throw exp	
	try exp **catch** $\{pexp_1, \ldots, pexp_n\}$	
	assert (exp, exp')	halt with error message exp' when not exp. exp
	(exp)	S
	var $lexp = exp$ **in** exp'	This is an internal node for compilation that de
	let $pat = exp$ **in** exp'	This is an internal node, used to distinguised so
	return_int (exp)	For internal use to embed into monad definition
	$value$	For internal use in interpreter to wrap pre-evalu
	constraint $n\_constraint$	

$lexp$	::=		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
$fexp$	::=		field expression
	|	$id = exp$	
$opt\_default$	::=		optional default value for indexed vector expressio
	|		
	|	; **default** $= exp$	
$pexp$	::=		pattern match
	|	$pat \rightarrow exp$	
	|	pat **when** $exp_1 \rightarrow exp$	
$tannot\_opt$	::=		optional type annotation for functions
	|		
	|	$typquant\ typ$	
rec_opt	::=		optional recursive annotation for functions
	|		non-recursive
	|	**rec**	recursive without termination measure

	$\mid$	$\{pat \rightarrow exp\}$ recursive with termination measure
effect_opt	::=	optional effect annotation for functions
	$\mid$	no effect annotation
	$\mid$	**effect** *effect*
pexp_funcl	::=	
	$\mid$	$pat = exp$
	$\mid$	$(pat\ \mathbf{when}\ exp_1) = exp$
funcl	::=	function clause
	$\mid$	*id pexp_funcl*
fundef	::=	function definition
	$\mid$	**function** *rec_opt tannot_opt effect_opt* $funcl_1$ **and** ... **and** $funcl_n$
mpat	::=	Mapping pattern. Mostly the same as normal patterns but c
	$\mid$	*lit*
	$\mid$	*id*
	$\mid$	$id(mpat_1, \ldots, mpat_n)$
	$\mid$	$[mpat_1, \ldots, mpat_n]$
	$\mid$	$mpat_1 @ \ldots @ mpat_n$
	$\mid$	$(mpat_1, \ldots, mpat_n)$
	$\mid$	$[\|\|mpat_1, \ldots, mpat_n\|\|]$
	$\mid$	$(mpat)$ S
	$\mid$	$mpat_1 :: mpat_2$
	$\mid$	$mpat_1 \uparrow\uparrow \ldots \uparrow\uparrow mpat_n$
	$\mid$	$mpat : typ$
	$\mid$	$mpat\ \mathbf{as}\ id$
mpexp	::=	
	$\mid$	*mpat*
	$\mid$	$mpat\ \mathbf{when}\ exp$
mapcl	::=	mapping clause (bidirectional pattern-match)
	$\mid$	$mpexp_1 \leftrightarrow mpexp_2$
	$\mid$	$mpexp \Rightarrow exp$
	$\mid$	$mpexp \mapsto exp$
mapdef	::=	mapping definition (bidirectional pattern-match function)
	$\mid$	**mapping** *id tannot_opt* $= \{mapcl_1, \ldots, mapcl_n\}$
letbind	::=	let binding
	$\mid$	**let** $pat = exp$ let, implicit type (*pat* must be total)
val_spec	::=	
	$\mid$	*val_spec_aux*

val_spec_aux	::=			value type specification
	|	**val** *typschm id*	S	specify the type of an upcoming definiti
	|	**val cast** *typschm id*	S	
	|	**val extern** *typschm id*	S	specify the type of an external function
	|	**val extern** *typschm id* = *string*	S	specify the type of a function from Lem
default_spec	::=			default kinding or typing assumption
	|	**default Order** *order*		
scattered_def	::=			scattered function and union type definiti
	|	**scattered function** *rec_opt tannot_opt effect_opt id*		scattered function definition header
	|	**function clause** *funcl*		scattered function definition clause
	|	**scattered typedef** *id* = **const union** *typquant*		scattered union definition header
	|	**union** *id* **member** *type_union*		scattered union definition member
	|	**scattered mapping** *id* : *tannot_opt*		
	|	**mapping clause** *id* = *mapcl*		
	|	**end** *id*		scattered definition end
reg_id	::=			
	|	*id*		
alias_spec	::=			register alias expression forms
	|	*reg_id*.*id*		
	|	*reg_id*[*exp*]		
	|	*reg_id*[*exp*..*exp*′]		
	|	*reg_id* : *reg_id*′		
dec_spec	::=			register declarations
	|	**register** *effect effect*′ *typ id*		
	|	**register configuration** *id* : *typ* = *exp*		
	|	**register alias** *id* = *alias_spec*		
	|	**register alias** *typ id* = *alias_spec*		
prec	::=			
	|	**infix**		
	|	**infixl**		
	|	**infixr**		
loop_measure	::=			
	|	*loop exp*		
def	::=			top-level definition
	|	*type_def*		type definition
	|	*fundef*		function definition

	|	*mapdef*	mapping definition
	|	*letbind*	value definition
	|	*val_spec*	top-level type cons
	|	**fix** *prec num id*	fixity declaration
	|	**overload** $id[id_1; \ldots; id_n]$	operator overload s
	|	*default_spec*	default kind and ty
	|	*scattered_def*	scattered function
	|	**termination_measure** *id pat* = *exp*	separate terminatio
	|	**termination_measure** *id* $loop\_measure_1, .., loop\_measure_n$	separate terminatio
	|	*dec_spec*	register declaration
	|	$fundef_1 .. fundef_n$	internal representa
	|	$\$string_1\ string_2\ l$	compiler directive
defs	::=		definition sequence
	|	$def_1 .. def_n$	
rv	::=		Constraint logic grou
	|	*num*	
	|	**true**	
	|	**false**	
	|	()	
	|	**bitstr**	
	|	(rv_1, rv_2)	
	|	$\dot{ctor}\ tid\ rv$	
	|	$\dot{ctor}\ tid\ b\ rv$	
	|	**usort** rv	
i	::=		RCL valuation
	|	ϵ	
	|	$x \rightarrow rv, i$	
b	::=		Base Type
	|	**int**	
	|	**bool**	
	|	*tid*	Type ID
	|	**unit**	
	|	**bvec**	Bit vectors
	|	$b_1 * b_2$	
	|	β	
	|	**bapp** *tid b*	
	|	$\|\tau\|_b$	M
	|	$b_1[b_2/\beta]$	M
τ	::=		Refined Type
	|	$\{x : b \| \phi\}$	bind x in ϕ
	|	$x : b[\phi]$	bind x in ϕ
	|	$\tau[v/x]$	M
	|	$\tau[b/\beta]$	M
	|	(τ)	S

A	::=			Dependent Function Type
	|	τ		
	|	$x : b[\phi] \to \tau$		
ce	::=			Expressions for constraints
	|	v		Value
	|	$ce_1 + ce_2$		Addition
	|	$va1 \leq va2$		Less than or equl
	|	$\mathbf{fst}\, ce$		Project first part of pair
	|	$\mathbf{snd}\, ce$		Project second part of pair
	|	$\mathbf{len}\, ce$		Length of vector
	|	$ce_1 @ ce_2$		Bit vector concat
	|	$ce[v/x]$	M	Substitution
	|	(ce)	S	
l	::=			Literals
	|	n		Numeric literal
	|	$\mathbf{T}$		true boolean literal
	|	$\mathbf{F}$		false boolean literal
	|	$()$		Unit value
	|	bin		Bit vector
v	::=			Values
	|	x		Immutable variable
	|	l		
	|	$v_1[v_2/x]$	M	Substitution
	|	(v)	S	
	|	(v_1, v_2)		Value pair
	|	$\dot{ctor}\ tid\ v$		Data constructor
	|	$\dot{ctor}\ tid[b]v$		Data constructor for polymorphic ty
def	::=			Definitions
	|	$\mathbf{val}\, f : (x : b[\phi]) \to \tau$	bind x in τ	
			bind x in ϕ	
	|	$\mathbf{val}\, \forall\, \beta. f : (x : b[\phi]) \to \tau$	bind x in τ	
			bind x in ϕ	
	|	$\mathbf{function}\, f(x) = s$	bind x in s	
	|	$\mathbf{function}\, f(x) = s$	bind x in s	
	|	$\mathbf{union}\, tid = \{\dot{ctor}_1 : \tau_1, \ldots, \dot{ctor}_n : \tau_n\}$		
	|	$\mathbf{union}\, tid = \forall\, \beta. \{\dot{ctor}_1 : \tau_1, \ldots, \dot{ctor}_n : \tau_n\}$		
p	::=			Program
	|	$def_1; \ldots; def_n; ; s$		
Γ	::=			Variable type context
	|	$\cdot$		Empty context
	|	$x : b[\phi]$		
	|	$\Gamma, x : b[\phi]$		
	|	(Γ)	S	

	|	Γ_1, Γ_2	
	|	$\Gamma[v/x]$	M

Φ	::=		Function context
	|	.	
	|	Φ, def	
	|	def	

Δ	::=		Mutable variables context
	|	.	
	|	Δ_1, Δ_2	
	|	(Δ)	S
	|	$\Delta, u : \tau$	
	|	$u : \tau$	

Θ	::=		Type defintions
	|	.	
	|	Θ, def	
	|	def	

B	::=		BCase type variable context
	|	.	
	|	B, β	
	|	β	

π	::=		Reduction Function Body Context
	|	.	
	|	$\pi, f : s$	

δ	::=		Reduction Local Store
	|	.	
	|	$\delta[u \mapsto v]$	

terminals	::=		
	|	**	**
	|	$\geq$	
	|	$\leq$	
	|	$\rightarrow$	
	|	$\leftrightarrow$	
	|	$\Longrightarrow$	==>
	|	$\cap$	
	|	$\uplus$	
	|	$\setminus$	
	|	$\notin$	
	|	$\subset$	
	|	$\neq$	
	|	$\emptyset$	
	|	$<$	
	|	$>$	

| | $\approx$
| | $\precapprox$
| | $\top$
| | $\vdash_t$
| | $\vdash_n$
| | $\vdash_e$
| | $\vdash_o$
| | $\vdash_c$
| | $'$
| | $\mapsto$
| | $\triangleright$
| | $\rightsquigarrow$
| | σ
| | $\Rightarrow$
| | $\_$
| | **effect**
| | ϵ
| | **consistent_increase**
| | **consistent_decrease**
| | $\equiv$
| | $\in$
| | $\restriction\sim$
| | $\sqsubseteq$
| | $\rightarrow$
| | $\vdash$
| | $\vdash^{\phi}$
| | $\vdash_a$
| | $\vdash$
| | $\vdash_{wf}$
| | $\dashv$
| | $\models$
| | $\Rightarrow$
| | $\Leftarrow$
| | $\vee$
| | $\wedge$
| | $\lesssim$
| | $\forall$
| | $\exists$
| | $\Longrightarrow$
| | $\longrightarrow$
| | $\rightsquigarrow$
| | $\in$
| | $\notin$
| | $\mapsto$
| | $\sim$

id ::= 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
	range	
	atom_bool	
	add_range	
	split_vector	
	vector_append	
	vector_access	
	vector_update	
	vector_subrange	
	fst	
	snd	
	len	
	$+$	
	$\leq$	

E	::=		
	|	ϵ	
	|	$E, id : typ$	
	|	E_{exp}	M
	|	E_{pat}	M
	|	E_{pexp}	M

M	::=		
	|	ϵ	
	|	$M, kid \rightarrow ce$	

e	::=		Expressions
	|	v	Value
	|	u	Mutable Variable
	|	$f\, v$	Function Application
	|	$f[b]v$	Polymorphic Function Application
	|	$v_1 + v_2$	Addition
	|	$v_1 \leq v_2$	Less than or equal
	|	$v_1 = v_2$	
	|	**fst** v	Project first part of pair
	|	**snd** v	Project second part of pair
	|	**len** e	
	|	$v_1 @ v_2$	

	$\mid$ $\mathbf{split}\, v_1\, v_2$		Split vector
	$\mid$ (e)	S	
	$\mid$ $e[v/x]$	M	Substitution
	$\mid$ $\mathbf{eqand}\, v_1\, v_2\, v_3$		
s	::=		Statement
	$\mid$ v		
	$\mid$ $\mathbf{let}\, x = e\, \mathbf{in}\, s$	bind x in s	Let binding
	$\mid$ $\mathbf{let}\ x : \tau = s_1\ \mathbf{in}\ s_2$	bind x in s_2	Let binding with type annotation
	$\mid$ $\mathbf{if}\, v\, \mathbf{then}\, s_1\, \mathbf{else}\, s_2$		If-then-else
	$\mid$ $s[v/x]$	M	Substitution
	$\mid$ $\mathbf{match}\, v\, \mathbf{of}\, \dot{ctor}_1\, x_1 \Rightarrow s_1, \ldots, \dot{ctor}_n\, x_n \Rightarrow s_n$		Match statement
	$\mid$ $\mathbf{var}\, u : \tau := v\, \mathbf{in}\, s$	bind u in s	Declaration and scoping of mutable
	$\mid$ $u := v$		Assignment to mutable variable
	$\mid$ $\mathbf{while}\, (s_1)\, \mathbf{do}\, \{s_2\}$		While loop
	$\mid$ $s_1; s_2$		Statement sequence
	$\mid$ $\mathbf{abort}$		
	$\mid$ $\mathbf{assert}\, \phi\, \mathbf{in}\, s$		
	$\mid$ (s)	S	
	$\mid$ $\{s\}$	S	
	$\mid$ $s[b/\beta]$	M	
	$\mid$ $L[s]$	M	
	$\mid$ $L[[s]]$	M	
	$\mid$ $\mathbf{switch}\, x\{lp_1 \Rightarrow s_1 \mid \ldots \mid lp_n \Rightarrow s_n\}$	M	
	$\mid$ $\mathbf{unpack}\, x\, \mathbf{into}\, x_1, \ldots, x_n\, \mathbf{in}\, s$	M	
γ	::=		Small context
	$\mid$ ϵ		
	$\mid$ $x : \tau$		
	$\mid$ γ_1, γ_2		
	$\mid$ $\gamma, x : \tau$		
L	::=		
	$\mid$ $\_\_$		
	$\mid$ $\mathbf{let}\, x = e\, \mathbf{in}\, L$		
	$\mid$ $\mathbf{let}\ x : \tau = s_1\ \mathbf{in}\ \_\_$		
	$\mid$ $L_1[L_2]$	M	
	$\mid$ $L_1 + \ldots + L_n$	M	
	$\mid$ (L)	S	
lp	::=		Literals for patterns. Augmenting wit
	$\mid$ l		
	$\mid$ $\_$		
	$\mid$ id		
π	::=		Pattern branch
	$\mid$ $pat_1 \ldots pat_n \Rightarrow exp$		patterns and associated term variab
	$\mid$ (π)	S	

Π	::=		Pattern matrix
	|	$\pi_1, .., \pi_n$	
	|	π, Π	
	|	$\cdot$	
ϕ	::=		Refinement Constraints - Quntifier fr
	|	$\top$	
	|	$\bot$	
	|	$\phi_1 \wedge \phi_2$	
	|	$\neg\phi$	
	|	$ce_1 = ce_2$	
	|	$ce_1 \leq ce_2$	
	|	(ϕ)	S
	|	$\phi[v/x]$	M
	|	$\phi_1 \Longrightarrow \phi_2$	
	|	$\phi_1 \wedge .. \wedge \phi_n$	
	|	$\phi[ce/x]$	
	|	$\phi[ce_1/x_1 .. ce_n/x_n]$	
mut	::=		
	|	**mutable**	
	|	**immutable**	
formula	::=		
	|	*judgement*	
	|	$formula_1 \quad .. \quad formula_n$	
	|	$x : b[\phi] \in \Gamma$	
	|	$u : \tau \in \Delta$	
	|	**union** $tid = \{\dot{ctor}_1 : \tau_1, \ldots, \dot{ctor}_n : \tau_n\} \in \Theta$	
	|	**union** $tid = \forall\, \beta.\{\dot{ctor}_1 : \tau_1, \ldots, \dot{ctor}_n : \tau_n\} \in \Theta$	
	|	$x \in \mathrm{dom}(\Gamma)$	
	|	**val** $f : (x : b[\phi]) \to \tau \notin \Phi$	
	|	**val** $f : (x : b[\phi]) \to \tau \in \Phi$	
	|	**val** $\forall\, \beta.f : (x : b[\phi]) \to \tau \in \Phi$	
	|	**function** $f(x) = s \notin \Phi$	
	|	**function** $f(x) = s \in \Phi$	
	|	$f \in \mathrm{dom}(\Phi)$	
	|	$u \in \mathrm{dom}(\Delta)$	
	|	$tid \notin \Phi$	
	|	$\dot{ctor} \notin \Phi$	
	|	$f \notin \Phi$	
	|	$f \notin \mathrm{dom}(\Phi)$	
	|	$u \notin \mathrm{dom}(\delta)$	
	|	$u \notin \mathrm{dom}(\Delta)$	
	|	$x \notin \mathrm{dom}(\Gamma)$	
	|	$tid \notin \mathrm{dom}(\Theta)$	
	|	**distinct** $\dot{ctor}_1 \ldots \dot{ctor}_n$	
	|	$\dot{ctor}_1 \ldots \dot{ctor}_n \notin \Theta$	
	|	$v_1 + v_2 = v$	
	|	$v_1 \leq v_2 = v$	

| $\mathbf{len}\, v_1 = v_2$
| $v_1 @ v_2 = v_3$
| $v_1 = \mathbf{split}\, v_2\, v_3$
| $f\, x = e$
| $x_1 = x_2$
| $x_1 \neq x_2$
| $x \# e$
| $x \# \Gamma$
| $x\, \mathbf{fresh}$
| $v = \delta(u)$
| $\delta' = \delta[u \mapsto v]$
| $\delta = u_1 \to v_1, .., u_n \to v_n$
| $\Delta = u_1 : \tau_1, .., u_n : \tau_n$
| $\beta \in B$
| $\forall i. \Theta; \Gamma \vdash i \wedge i \models \Gamma \longrightarrow i \models \phi$
| $rv = i(x)$
| $rv = rv_1 + rv_2$
| $rv = rv_1 \leq rv_2$
| $rv = rv_1 @ rv_2$
| $rv = \mathbf{len}\, rv'$
| $rv = (rv_1 = rv_2)$
| $rv = rv_1 \vee rv_2$
| $rv = rv_1 \wedge rv_2$
| $rv = rv_1 \Longrightarrow rv_2$
| $rv = \sim rv_1$
| $id \sim x$
| $id \sim u$
| $E \vdash id \rightsquigarrow \dot{ctor}, tid$
| $id \sim tid$
| $lp \notin lp_1 .. lp_n$
| $id \in E.mutable$
| $id \in E.immutable$
| $id \in E.enum$
| $id \in E.ctor$
| $id/mut : typ \in E$
| $id/\mathbf{register} : typ \in E$
| $id/\mathbf{enum} : typ \in E$
| $id/mut \notin E$
| $\mathbf{fresh}\, x$
| $\mathbf{fresh}\, x_1 .. x_n$
| $num = \mathbf{is\_constant}\, ce$
| $quant\_item_1, \ldots, quant\_item_n \rightsquigarrow kinded\_id_1 .. kinded\_id_m, n\_constraint$
| $b \in \{\mathbf{int}, \mathbf{bool}\}$
| $\mathbf{is\_ctor}\, b$
| $b = (b_1, \ldots, b_n)$
| $\mathbf{fresh}\, x_1 .. x_n$
| $pat_1 .. pat_n = \mathbf{duplicate}\, pat\, b_1 .. b_m$
| $kind_1 = kind_2$
| $kind_1 \neq kind_2$
| $kid = ka$

	$M' = M, ce, kinded\_id_1 \,..\, kinded\_id_m$	
	$\textbf{is\_kid\_map}\, M', b, ce, kinded\_id_1 \,..\, kinded\_id_m$	
	$ce = M(kid)$	
	$id_1 \,...\, id_n \rightsquigarrow f$	
	$E \vdash \textbf{inst\_of}\, id(exp_1, \,..\, , exp_n) \rightsquigarrow x; L$	
	$L_1 = L_2$	
	$L_5 = \textbf{let}\, x_4 = \textbf{update\_vector\_range}\, x\, x_1\, x_2\, x_3\, \textbf{in}\, \_\_$	

rcl ::=

	$[\![l]\!] \sim rv$	
	$i[\![v]\!] \sim rv$	
	$i[\![ce]\!] \sim rv$	
	$i[\![\phi]\!] \sim rv$	
	$i \models \phi$	
	$i \models \Gamma$	
	$\Theta \vdash_{wf} rv : b$	
	$\Theta; \Gamma \vdash i$	
	$\Theta; B; \Gamma \models \phi$	

wf_check ::=

	$\vdash_{wf} \Theta$	Wellformedness for type def
	$\Theta; B \vdash_{wf} b$	Wellformedness for base-typ
	$\Theta \vdash_{wf} \Phi$	Wellformedness for function
	$\Theta; B \vdash_{wf} \Gamma$	Wellformedness for immutab
	$\Theta; B; \Gamma \vdash_{wf} \Delta$	Wellformedness for mutable
	$\Theta; B; \Gamma \vdash_{wf} v : b$	WF for values
	$\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} e : b$	WF for expressions
	$\Theta; B; \Gamma \vdash_{wf} \phi$	WF for constraints
	$\Theta; B; \Gamma \vdash_{wf} \tau$	WF for types
	$\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s : b$	WF for statements

extension ::=

	$\Theta; B \vdash \Gamma_1 \sqsubseteq \Gamma_2$	Γ_2 is an extension of Γ_1
	$\Theta; B; \Gamma \vdash \Delta_2 \sqsubseteq \Delta_1$	Δ_1 is an extension of Δ_2

subtype_anf ::=

	$\Theta; B; \Gamma \vdash \tau_1 \precapprox \tau_2$	Subtyping

typing ::=

	$\vdash l \Rightarrow \tau$	Type synthesis for literals.
	$\Theta; B; \Gamma \vdash v \Rightarrow \tau$	Type synthesis. Infer that t
	$\Theta; B; \Gamma \vdash v \leq \tau$	Check that type of v is τ
	$\Theta; \Phi; B; \Gamma; \Delta \vdash e \Rightarrow \tau$	Infer that type of e is τ
	$\Theta; \Phi; B; \Gamma; \Delta \vdash e \leq \tau$	Check that type of e is τ
	$\Theta; \Phi; B; \Gamma; \Delta \vdash s \leq \tau$	Check that type of s is τ
	$\Theta_1; \Phi_1 \vdash def_1 \,..\, def_n \rightsquigarrow \Theta_2; \Phi_2$	
	$\vdash p$	
	$\Theta \vdash \Delta \sim \delta$	
	$\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau$	Program state typing judge

reduction	::=		
	|	$\Phi \vdash \langle \delta, s_1 \rangle \rightarrow \langle \delta', s_2 \rangle$	One step reduction
	|	$\Phi \vdash \langle \delta_1, s_1 \rangle \xrightarrow{*} \langle \delta_2, s_2 \rangle$	Multi-step reduction
check_config	::=		
	|	$\Theta \vdash \delta \sim \Delta$	
	|	$\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau$	
record	::=		
	|	$E \vdash$ **pack_record** $\tau\, x\, id_1 = x_1 \ldots id_n = x_n \rightsquigarrow L$	
	|	$E \vdash$ **unpack_field** $\tau\, x\, x'\, id \rightsquigarrow L$	
	|	$E \vdash$ **update_record** $\tau\, x\, x'\, id_1 = x_1 \ldots id_n = x_n \rightsquigarrow L$	
wf_l	::=		
	|	$\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} L : \gamma$	WF for let-context
convert_typ	::=		
	|	$typquant \rightsquigarrow kinded\_id_1 \,..\, kinded\_id_m, n\_constraint$	
	|	$E \vdash typ \rightsquigarrow \tau$	
	|	$E; M \vdash typ\_arg \rightsquigarrow \phi$	
	|	$E; M \vdash typ\_arg \rightsquigarrow ce$	
	|	$E; M \vdash typ; ce \rightsquigarrow b; \phi$	
	|	$E; M \vdash n\_constraint \rightsquigarrow \phi$	
	|	$E; M \vdash nexp \rightsquigarrow ce$	
convert_exp	::=		
	|	$lit \rightsquigarrow lp$	
	|	$lit \rightsquigarrow l$	
	|	$E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta$	
	|	$E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta / \gamma \vdash x; L : \tau$	
	|	$E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau$	
	|	$E \vdash \Pi : b_1 / x_1 \,..\, b_n / x_n \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau$	Convert match branch
pattern_expansion	::=		
	|	$E \vdash x_1; x_2; pat_1, \ldots, pat_n \rightsquigarrow L; x_3$	
	|	$E \vdash \Pi \rightsquigarrow \Pi_1; lp_1 \,\|\|\, .. \,\|\|\, \Pi_n; lp_n$	
	|	$E \vdash \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\, b_1\, x_1 \,\|\|\, .. \,\|\|\, \Pi_n; \dot{ctor}_n\, b_n\, x_n$	
	|	$E \vdash \Pi : b \rightsquigarrow \Pi'; b_1 / x_1 \,..\, b_n / x_n$	
convert_defs	::=		
	|	$E \vdash funcl_1$ **and** $\ldots$ **and** $funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def$	
	|	$E \vdash def \rightsquigarrow \Theta; \Phi; \Delta \vdash def_1, \ldots, def_n$	
	|	$E \vdash def_1 \,..\, def_n \rightsquigarrow \Theta; \Phi \vdash def_1 \,..\, def_m$	
judgement	::=		
	|	*rcl*	
	|	*wf_check*	
	|	*extension*	
	|	*subtype_anf*	

| *typing*
| *reduction*
| *check_config*
| *record*
| *wf_l*
| *convert_typ*
| *convert_exp*
| *pattern_expansion*
| *convert_defs*

user_syntax ::=
| *n*
| *num*
| *nat*
| *hex*
| *bin*
| *string*
| *regexp*
| *real*
| *value*
| *x*
| *ix*
| q
| $\dot{ctor}$
| *x*
| *bit*
| *u*
| β
| *tid*
| *l*
| *annot*
| *kid*
| *kind*
| *nexp*
| *order*
| *base_effect*
| *effect*
| *typ*
| *typ_arg*
| *n_constraint*
| *kinded_id*
| *quant_item*
| *typquant*
| *typschm*
| *type_def*
| *type_def_aux*
| *type_union*
| *index_range*
| *lit*
| $;^?$

| typ_pat
| pat
| $loop$
| $internal\_loop\_measure$
| exp
| $lexp$
| $fexp$
| $opt\_default$
| $pexp$
| $tannot\_opt$
| rec_opt
| $effect\_opt$
| $pexp\_funcl$
| $funcl$
| $fundef$
| $mpat$
| $mpexp$
| $mapcl$
| $mapdef$
| $letbind$
| val_spec
| val_spec_aux
| $default\_spec$
| $scattered\_def$
| reg_id
| $alias\_spec$
| dec_spec
| $prec$
| $loop\_measure$
| def
| $defs$
| rv
| i
| b
| τ
| A
| ce
| l
| v
| def
| p
| Γ
| Φ
| Δ
| Θ
| B
| π
| δ
| $terminals$
| id

| E
| M
| e
| s
| γ
| L
| lp
| π
| Π
| ϕ
| mut
| $formula$

1 Syntax

The syntax ...

2 MiniSail type system

2.1 Refinement constraint logic

$\boxed{[\![l]\!] \sim rv}$

$$\frac{}{[\![n]\!] \sim num} \text{ EVAL\_LIT\_NUM}$$

$$\frac{}{[\![\mathbf{T}]\!] \sim \mathbf{true}} \text{ EVAL\_LIT\_TRUE}$$

$$\frac{}{[\![\mathbf{F}]\!] \sim \mathbf{false}} \text{ EVAL\_LIT\_FALSE}$$

$$\frac{}{[\![()]\!] \sim ()} \text{ EVAL\_LIT\_UNIT}$$

$\boxed{i[\![v]\!] \sim rv}$

$$\frac{[\![l]\!] \sim rv}{i[\![l]\!] \sim rv} \text{ EVAL\_V\_LIT}$$

$$\frac{rv = i(x)}{i[\![x]\!] \sim rv} \text{ EVAL\_V\_VAR}$$

$$\frac{\begin{array}{c} i[\![v_1]\!] \sim rv_1 \\ i[\![v_2]\!] \sim rv_2 \end{array}}{i[\![(v_1, v_2)]\!] \sim (rv_1, rv_2)} \text{ EVAL\_V\_PAIR}$$

$$\frac{i[\![v]\!] \sim rv}{i[\![\dot{ctor}\ tid\ v]\!] \sim \dot{ctor}\ tid\ rv} \text{ EVAL\_V\_CONS}$$

$$\frac{i[\![v]\!] \sim rv}{i[\![\dot{ctor}\ tid[b]v]\!] \sim \dot{ctor}\ tid\ b\ rv} \text{ EVAL\_V\_CONSP}$$

$\boxed{i[\![ce]\!] \sim rv}$

$$\frac{i[\![v]\!] \sim rv}{i[\![v]\!] \sim rv} \quad \text{EVAL\_CE\_VAL}$$

$$\frac{\begin{array}{c} i[\![v_1]\!] \sim rv_1 \\ i[\![v_2]\!] \sim rv_2 \\ rv = rv_1 + rv_2 \end{array}}{i[\![v_1 + v_2]\!] \sim rv} \quad \text{EVAL\_CE\_PLUS}$$

$$\frac{\begin{array}{c} i[\![v_1]\!] \sim rv_1 \\ i[\![v_2]\!] \sim rv_2 \\ rv = rv_1 \leq rv_2 \end{array}}{i[\![va1 \leq va2]\!] \sim rv} \quad \text{EVAL\_CE\_LEQ}$$

$$\frac{i[\![v_1]\!] \sim rv_1}{i[\![\mathbf{fst}\,(v_1, v_2)]\!] \sim rv_1} \quad \text{EVAL\_CE\_FST}$$

$$\frac{i[\![v_2]\!] \sim rv_2}{i[\![\mathbf{snd}\,(v_1, v_2)]\!] \sim rv_2} \quad \text{EVAL\_CE\_SND}$$

$$\frac{\begin{array}{c} i[\![v_1]\!] \sim rv_1 \\ i[\![v_2]\!] \sim rv_2 \\ rv = rv_1 @ rv_2 \end{array}}{i[\![v_1 @ v_2]\!] \sim rv} \quad \text{EVAL\_CE\_CONCAT}$$

$$\frac{\begin{array}{c} i[\![v]\!] \sim rv' \\ rv = \mathbf{len}\, rv' \end{array}}{i[\![\mathbf{len}\, v_1]\!] \sim rv} \quad \text{EVAL\_CE\_LEN}$$

$\boxed{i[\![\phi]\!] \sim rv}$

$$\frac{\begin{array}{c} i[\![ce_1]\!] \sim rv_1 \\ i[\![ce_2]\!] \sim rv_2 \\ rv = (rv_1 = rv_2) \end{array}}{i[\![ce_1 = ce_2]\!] \sim rv} \quad \text{EVAL\_C\_EQ}$$

$$\frac{\begin{array}{c} i[\![\phi_1]\!] \sim rv_1 \\ i[\![\phi_2]\!] \sim rv_2 \\ rv = rv_1 \wedge rv_2 \end{array}}{i[\![\phi_1 \wedge \phi_2]\!] \sim rv} \quad \text{EVAL\_C\_AND}$$

$$\frac{\begin{array}{c} i[\![\phi]\!] \sim rv' \\ rv = \sim rv' \end{array}}{i[\![\neg\phi]\!] \sim rv} \quad \text{EVAL\_C\_NOT}$$

$$\frac{\begin{array}{c} i[\![\phi_1]\!] \sim rv_1 \\ i[\![\phi_2]\!] \sim rv_2 \\ rv = rv_1 \Longrightarrow rv_2 \end{array}}{i[\![\phi_1 \Longrightarrow \phi_2]\!] \sim rv} \quad \text{EVAL\_C\_IMP}$$

$\boxed{i \models \phi}$

$$\frac{i[\![\phi]\!] \sim \mathbf{true}}{i \models \phi} \quad \text{SATIS\_CA\_CA}$$

$\boxed{i \models \Gamma}$

$$\frac{}{i \models \cdot}\ \text{SATIS\_G\_NIL}$$

$$\frac{i \models \Gamma \quad i \models \phi}{i \models \Gamma, x : b[\phi]}\ \text{SATIS\_G\_CONS}$$

$\boxed{\Theta \vdash_{wf} rv : b}$

$$\frac{}{\Theta \vdash_{wf} num : \mathbf{int}}\ \text{WF\_RCL\_V\_INT}$$

$$\frac{}{\Theta \vdash_{wf} \mathbf{true} : \mathbf{bool}}\ \text{WF\_RCL\_V\_TRUE}$$

$$\frac{}{\Theta \vdash_{wf} \mathbf{false} : \mathbf{bool}}\ \text{WF\_RCL\_V\_FALSE}$$

$$\frac{}{\Theta \vdash_{wf} () : \mathbf{unit}}\ \text{WF\_RCL\_V\_UNIT}$$

$$\frac{}{\Theta \vdash_{wf} \mathbf{bitstr} : \mathbf{bvec}}\ \text{WF\_RCL\_V\_BVEC}$$

$$\frac{\Theta \vdash_{wf} rv_1 : b_1 \quad \Theta \vdash_{wf} rv_2 : b_2}{\Theta \vdash_{wf} (rv_1, rv_2) : b_1 * b_2}\ \text{WF\_RCL\_V\_PAIR}$$

$$\frac{\Theta \vdash_{wf} rv : b \quad \mathbf{union}\, tid = \{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta}{\Theta \vdash_{wf} \dot{ctor}_j\, tid\, rv : tid}\ \text{WF\_RCL\_V\_CONS}$$

$$\frac{\Theta \vdash_{wf} rv : |\tau_j|_b[b_2/\beta] \quad \mathbf{union}\, tid = \forall\, \beta.\{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta}{\Theta \vdash_{wf} \dot{ctor}_j\, tid\, b_2\, rv : \mathbf{bapp}\, tid\, b_2}\ \text{WF\_RCL\_V\_CONSP}$$

$$\frac{}{\Theta \vdash_{wf} \mathbf{usort}\, rv : \beta}\ \text{WF\_RCL\_V\_BOXED}$$

$\boxed{\Theta; \Gamma \vdash i}$

$$\frac{}{\Theta; \cdot \vdash i}\ \text{WF\_VAL\_EMPTY}$$

$$\frac{rv = i(x) \quad \Theta \vdash_{wf} rv : b}{\Theta; \Gamma, x : b[\phi] \vdash i}\ \text{WF\_VAL\_CONS}$$

$\boxed{\Theta; B; \Gamma \models \phi}$

$$\frac{\Theta; B; \Gamma \vdash_{wf} \phi \quad \forall i. \Theta; \Gamma \vdash i \wedge i \models \Gamma \longrightarrow i \models \phi}{\Theta; B; \Gamma \models \phi}\ \text{VALID\_VALID}$$

2.2 Wellformedness

$\boxed{\vdash_{wf} \Theta}$ Wellformedness for type definition context

$$\frac{}{\vdash_{wf} \cdot} \quad \text{THETA\_BEMPTY}$$

$$\frac{\begin{array}{l} tid \notin \mathrm{dom}(\Theta) \\ \textbf{distinct}\, \dot{ctor}_1 \ldots \dot{ctor}_n \\ \dot{ctor}_1 \ldots \dot{ctor}_n \notin \Theta \end{array}}{\vdash_{wf} \Theta, \textbf{union}\, tid = \{\dot{ctor}_1 : \tau_1, \ldots, \dot{ctor}_n : \tau_n\}} \quad \text{THETA\_BUNION}$$

$\boxed{\Theta; B \vdash_{wf} b}$ Wellformedness for base-type

$$\frac{\vdash_{wf} \Theta}{\Theta; B \vdash_{wf} \textbf{bool}} \quad \text{WF\_B\_BOOL}$$

$$\frac{\vdash_{wf} \Theta}{\Theta; B \vdash_{wf} \textbf{int}} \quad \text{WF\_B\_INT}$$

$$\frac{\vdash_{wf} \Theta}{\Theta; B \vdash_{wf} \textbf{unit}} \quad \text{WF\_B\_UNIT}$$

$$\frac{\vdash_{wf} \Theta}{\Theta; B \vdash_{wf} \textbf{bvec}} \quad \text{WF\_B\_BVEC}$$

$$\frac{\begin{array}{l} \Theta; B \vdash_{wf} b_1 \\ \Theta; B \vdash_{wf} b_2 \end{array}}{\Theta; B \vdash_{wf} b_1 * b_2} \quad \text{WF\_B\_PAIR}$$

$$\frac{\begin{array}{l} \vdash_{wf} \Theta \\ \textbf{union}\, tid = \{\dot{ctor}_1 : \tau_1, .., \dot{ctor}_n : \tau_n\} \in \Theta \end{array}}{\Theta; B \vdash_{wf} tid} \quad \text{WF\_B\_TID}$$

$$\frac{\beta \in B}{\Theta; B \vdash_{wf} \beta} \quad \text{WF\_B\_BVR}$$

$\boxed{\Theta \vdash_{wf} \Phi}$ Wellformedness for function definition context

$$\frac{\begin{array}{l} f \notin \mathrm{dom}(\Phi) \\ \Theta; \cdot, \beta \vdash_{wf} b \\ \Theta; \cdot, \beta \vdash_{wf} x : b[\phi] \\ \Theta; \cdot, \beta; x : b[\phi] \vdash_{wf} \tau \end{array}}{\Theta \vdash_{wf} \Phi, \textbf{val} \forall \beta. f : (x : b[\phi]) \to \tau} \quad \text{WF\_P\_VALSPEC\_POLY}$$

$$\frac{\begin{array}{l} f \notin \mathrm{dom}(\Phi) \\ \Theta; \cdot \vdash_{wf} b \\ \Theta; \cdot \vdash_{wf} x : b[\phi] \\ \Theta; \cdot; x : b[\phi] \vdash_{wf} \tau \end{array}}{\Theta \vdash_{wf} \Phi, \textbf{val}\, f : (x : b[\phi]) \to \tau} \quad \text{WF\_P\_VALSPEC}$$

$$\frac{\vdash_{wf} \Theta}{\Theta \vdash_{wf} \cdot} \quad \text{WF\_P\_EMPTY}$$

$\boxed{\Theta; B \vdash_{wf} \Gamma}$ Wellformedness for immutable variable context

$$\frac{\vdash_{wf} \Theta}{\Theta; B \vdash_{wf} \cdot} \quad \text{WF\_G\_EMPTY}$$

$$\frac{\begin{array}{l}\Theta; B \vdash_{wf} \Gamma \\ \Theta; B \vdash_{wf} b \\ \Theta; B; \Gamma, x : b[\top] \vdash_{wf} \phi \\ x \notin \mathrm{dom}(\Gamma)\end{array}}{\Theta; B \vdash_{wf} \Gamma, x : b[\phi]} \quad \text{WF\_G\_CONS}$$

$$\frac{\begin{array}{l}\Theta; B \vdash_{wf} \Gamma \\ \Theta; B \vdash_{wf} b \\ x \notin \mathrm{dom}(\Gamma)\end{array}}{\Theta; B \vdash_{wf} \Gamma, x : b[\top]} \quad \text{WF\_G\_CONS\_TRUE}$$

$$\frac{\begin{array}{l}\Theta; B \vdash_{wf} \Gamma \\ \Theta; B \vdash_{wf} b \\ x \notin \mathrm{dom}(\Gamma)\end{array}}{\Theta; B \vdash_{wf} \Gamma, x : b[\bot]} \quad \text{WF\_G\_CONS\_FALSE}$$

$\boxed{\Theta; B; \Gamma \vdash_{wf} \Delta}$ Wellformedness for mutable variable context

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B; \Gamma \vdash_{wf} \cdot} \quad \text{WF\_D\_EMPTY}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} \tau \\ u \notin \mathrm{dom}(\Delta)\end{array}}{\Theta; B; \Gamma \vdash_{wf} \Delta, u : \tau} \quad \text{WF\_D\_CONS}$$

$\boxed{\Theta; B; \Gamma \vdash_{wf} v : b}$ WF for values

$$\frac{\begin{array}{l}\Theta; B \vdash_{wf} \Gamma \\ x : b[\phi] \in \Gamma\end{array}}{\Theta; B; \Gamma \vdash_{wf} x : b} \quad \text{WF\_V\_VAR}$$

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B; \Gamma \vdash_{wf} n : \mathbf{int}} \quad \text{WF\_V\_NUM}$$

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B; \Gamma \vdash_{wf} \mathbf{T} : \mathbf{bool}} \quad \text{WF\_V\_TRUE}$$

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B; \Gamma \vdash_{wf} \mathbf{F} : \mathbf{bool}} \quad \text{WF\_V\_FALSE}$$

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B; \Gamma \vdash_{wf} () : \mathbf{unit}} \quad \text{WF\_V\_UNIT}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} v : |\tau_j|_b \\ \mathbf{union}\, tid = \{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta\end{array}}{\Theta; B; \Gamma \vdash_{wf} \dot{ctor}_j\ tid\ v : tid} \quad \text{WF\_V\_CONS}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} v : |\tau_j|_b[b_2/\beta] \\ \Theta; B \vdash_{wf} b_2 \\ \mathbf{union}\, tid = \forall\, \beta.\{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta\end{array}}{\Theta; B; \Gamma \vdash_{wf} \dot{ctor}_j\ tid[b_2]v : \mathbf{bapp}\ tid\ b_2} \quad \text{WF\_V\_CONSP}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} v_1 : b_1 \\ \Theta; B; \Gamma \vdash_{wf} v_2 : b_2\end{array}}{\Theta; B; \Gamma \vdash_{wf} (v_1, v_2) : b_1 * b_2} \quad \text{WF\_V\_PAIR}$$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} e : b}$ WF for expressions

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} v : b \\ \mathbf{val}\, f : (x : b[\phi]) \to \tau \in \Phi\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} f\, v : |\tau|_b} \text{ WF\_E\_APP}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} v : b_1[b_2/\beta] \\ \mathbf{val}\, \forall\, \beta. f : (x : b_1[\phi]) \to \tau \in \Phi\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} f[b_2]v : |\tau|_b[b_2/\beta]} \text{ WF\_E\_APP\_POLY}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v_1 : \mathbf{int} \\ \Theta; B; \Gamma \vdash_{wf} v_2 : \mathbf{int}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} v_1 + v_2 : \mathbf{int}} \text{ WF\_E\_PLUS}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v_1 : \mathbf{int} \\ \Theta; B; \Gamma \vdash_{wf} v_2 : \mathbf{int}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} v_1 \leq v_2 : \mathbf{bool}} \text{ WF\_E\_LEQ}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v : b_1 * b_2\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{fst}\, v : b_1} \text{ WF\_E\_FST}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v : b_1 * b_2\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{snd}\, v : b_2} \text{ WF\_E\_SND}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v : \mathbf{bvec}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{len}\, v : \mathbf{int}} \text{ WF\_E\_LEN}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v_1 : \mathbf{bvec} \\ \Theta; B; \Gamma \vdash_{wf} v_2 : \mathbf{bvec}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} v_1 @ v_2 : \mathbf{bvec}} \text{ WF\_E\_CONCAT}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v_1 : \mathbf{int} \\ \Theta; B; \Gamma \vdash_{wf} v_2 : \mathbf{bvec}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{split}\, v_1\, v_2 : \mathbf{bvec} * \mathbf{bvec}} \text{ WF\_E\_SPLIT}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ u : \tau \in \Delta\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} u : |\tau|_b} \text{ WF\_E\_MVAR}$$

$\boxed{\Theta; B; \Gamma \vdash_{wf} \phi}$ WF for constraints

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \phi_1 \\ \Theta; B; \Gamma \vdash_{wf} \phi_2\end{array}}{\Theta; B; \Gamma \vdash_{wf} \phi_1 \wedge \phi_2} \text{ WF\_C\_CONJ}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \phi_1 \\ \Theta; B; \Gamma \vdash_{wf} \phi_2\end{array}}{\Theta; B; \Gamma \vdash_{wf} \phi_1 \Longrightarrow \phi_2} \text{ WF\_C\_IMP}$$

$$\frac{\begin{array}{l}\Theta; \cdot; B; \Gamma; \cdot \vdash_{wf} e_1 : b \\ \Theta; \cdot; B; \Gamma; \cdot \vdash_{wf} e_2 : b\end{array}}{\Theta; B; \Gamma \vdash_{wf} ce_1 = ce_2} \text{ WF\_C\_EQ}$$

$\boxed{\Theta; B; \Gamma \vdash_{wf} \tau}$ WF for types

$$\frac{\Theta; B; \Gamma, z : b[\top] \vdash_{wf} \phi}{\Theta; B; \Gamma \vdash_{wf} \{z : b | \phi\}} \text{ WF\_T\_TAU}$$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s : b}$ WF for statements

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash_{wf} v : b\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} v : b} \text{ WF\_S\_VAL}$$

$$\frac{\begin{array}{l}u \notin \mathrm{dom}(\Delta) \\ \Theta; B; \Gamma \vdash_{wf} v : b_1 \\ \Theta; \Phi; B; \Gamma; \Delta, u : \tau \vdash_{wf} s : b_2\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{var}\, u : \tau := v \,\mathbf{in}\, s : b_2} \text{ WF\_S\_VAR}$$

$$\frac{\begin{array}{l}\Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ u : \{z : b | \phi\} \in \Delta \\ \Theta; B; \Gamma \vdash_{wf} v : b\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} u := v : \mathbf{unit}} \text{ WF\_S\_ASSIGN}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} v : \mathbf{bool} \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_1 : b \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_2 : b\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{if}\, v \,\mathbf{then}\, s_1 \,\mathbf{else}\, s_2 : b} \text{ WF\_S\_IF}$$

$$\frac{\begin{array}{l}x \# \Gamma \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} e : b_1 \\ \Theta; \Phi; B; \Gamma, x : b_1[\phi]; \Delta \vdash_{wf} s : b_2\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{let}\, x = e \,\mathbf{in}\, s : b_2} \text{ WF\_S\_LET}$$

$$\frac{\begin{array}{l}x \# \Gamma \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_1 : b_1 \\ \Theta; \Phi; B; \Gamma, x : b_1[\top]; \Delta \vdash_{wf} s_2 : b_2 \\ \Theta; B; \Gamma \vdash_{wf} \{z : b_1 | \phi\}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{let}\ x : \{z : b_1 | \phi\} = s_1\ \mathbf{in}\ s_2 : b_2} \text{ WF\_S\_LET2}$$

$$\frac{\begin{array}{l}\mathbf{union}\ tid = \{\ \overline{\dot{ctor}_i : \{z_i : b_i | \phi_i\}}^{\,i}\ \} \in \Theta \\ \Theta; B; \Gamma \vdash_{wf} v : tid \\ \overline{\Theta; \Phi; B; \Gamma, x_i : b_i[v = \dot{ctor}_i\ tid\ x_i \wedge \phi_i[x_i/z_i]]; \Delta \vdash_{wf} s_i : b}^{\,i}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{match}\, v\, \mathbf{of}\, \overline{\dot{ctor}_i\, x_i \Rightarrow s_i}^{\,i} : b}\ \text{WF\_S\_MATCH}$$

$$\frac{\begin{array}{c}\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_1 : \mathbf{bool} \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_2 : \mathbf{unit}\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{while}\,(s_1)\,\mathbf{do}\,\{s_2\} : \mathbf{unit}}\ \text{WF\_S\_WHILE}$$

$$\frac{\begin{array}{c}\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_1 : \mathbf{unit} \\ \Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_2 : b\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s_1; s_2 : b}\ \text{WF\_S\_SEQ}$$

$$\frac{\begin{array}{l}x \# \Gamma \\ \Theta; B; \Gamma \vdash_{wf} \phi \\ \Theta; \Phi; B; \Gamma, x : \mathbf{bool}[\phi]; \Delta \vdash_{wf} s : b\end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \mathbf{assert}\,\phi\,\mathbf{in}\,s : b}\ \text{WF\_S\_ASSERT}$$

$\boxed{\Theta; B \vdash \Gamma_1 \sqsubseteq \Gamma_2}$ Γ_2 is an extension of Γ_1

$$\frac{\Theta; B \vdash_{wf} \Gamma}{\Theta; B \vdash \Gamma \sqsubseteq \Gamma}\ \text{EXTEND\_G\_REFL}$$

$$\frac{\begin{array}{l}\Theta; B \vdash \Gamma_3 \sqsubseteq \Gamma_1, \Gamma_2 \\ x \notin \mathrm{dom}(\Gamma_1, \Gamma_2) \\ \Theta; B \vdash_{wf} \Gamma, x : b[\phi]\end{array}}{\Theta; B \vdash \Gamma_3 \sqsubseteq \Gamma_1, (\Gamma_2, x : b[\phi])}\ \text{EXTEND\_G\_INSERT}$$

$\boxed{\Theta; B; \Gamma \vdash \Delta_2 \sqsubseteq \Delta_1}$ Δ_1 is an extension of Δ_2

$$\frac{\Theta; B; \Gamma \vdash_{wf} \Delta}{\Theta; B; \Gamma \vdash \Delta \sqsubseteq \Delta}\ \text{EXTEND\_D\_REFL}$$

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash \Delta_3 \sqsubseteq \Delta_1, \Delta_2 \\ u \notin \mathrm{dom}(\Delta_1, \Delta_2) \\ \Theta; B; \Gamma \vdash_{wf} \tau\end{array}}{\Theta; B; \Gamma \vdash \Delta_3 \sqsubseteq \Delta_1, (\Delta_2, u : \tau)}\ \text{EXTEND\_D\_INSERT}$$

2.3 Subtyping

$\boxed{\Theta; B; \Gamma \vdash \tau_1 \precsim \tau_2}$ Subtyping

$$\frac{\begin{array}{l}\Theta; B; \Gamma \vdash_{wf} \{z_1 : b | \phi_1\} \\ \Theta; B; \Gamma \vdash_{wf} \{z_2 : b | \phi_2\} \\ \Theta; B; \Gamma, z_3 : b[\phi_1[z_3/z_1]] \models \phi_2[z_3/z_1]\end{array}}{\Theta; B; \Gamma \vdash \{z_1 : b | \phi_1\} \precsim \{z_2 : b | \phi_2\}}\ \text{SUBTYPE\_ANF\_SUBTYPE}$$

2.4 Typing

$\boxed{\vdash l \Rightarrow \tau}$ Type synthesis for literals. Infer that type of l is τ

$$\frac{}{\vdash () \Rightarrow \{z : \mathbf{unit} | z = ()\}} \text{ INFER\_L\_UNIT}$$

$$\frac{}{\vdash \mathbf{T} \Rightarrow \{z : \mathbf{bool} | z = \mathbf{T}\}} \text{ INFER\_L\_TRUE}$$

$$\frac{}{\vdash \mathbf{F} \Rightarrow \{z : \mathbf{bool} | z = \mathbf{F}\}} \text{ INFER\_L\_FALSE}$$

$$\frac{}{\vdash n \Rightarrow \{z : \mathbf{int} | z = n\}} \text{ INFER\_L\_NUM}$$

$$\frac{}{\vdash bin \Rightarrow \{z : \mathbf{bvec} | z = bin\}} \text{ INFER\_L\_BVEC}$$

$\boxed{\Theta; B; \Gamma \vdash v \Rightarrow \tau}$ Type synthesis. Infer that type of v is τ

$$\frac{\begin{array}{c} z \# \Gamma \\ \Theta; B \vdash_{wf} \Gamma \\ x : b[\phi] \in \Gamma \end{array}}{\Theta; B; \Gamma \vdash x \Rightarrow \{z : b | z = x\}} \text{ INFER\_V\_ANF\_VAR}$$

$$\frac{\begin{array}{c} \vdash l \Rightarrow \tau \\ \Theta; B \vdash_{wf} \Gamma \end{array}}{\Theta; B; \Gamma \vdash l \Rightarrow \tau} \text{ INFER\_V\_ANF\_LIT}$$

$$\frac{\begin{array}{c} z \# \Gamma \\ \Theta; B; \Gamma \vdash v_1 \Rightarrow \{z_1 : b_1 | \phi_1\} \\ \Theta; B; \Gamma \vdash v_2 \Rightarrow \{z_2 : b_2 | \phi_2\} \end{array}}{\Theta; B; \Gamma \vdash (v_1, v_2) \Rightarrow \{z : b_1 * b_2 | z = (v_1, v_2)\}} \text{ INFER\_V\_ANF\_PAIR}$$

$$\frac{\begin{array}{c} z \# \Gamma \\ \mathbf{union}\, tid = \{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta \\ \Theta; B; \Gamma \vdash v \leq \tau \end{array}}{\Theta; B; \Gamma \vdash \dot{ctor}_j\ tid\ v \Rightarrow \{z : tid | z = \dot{ctor}_j\ tid\ v\}} \text{ INFER\_V\_ANF\_DATA\_CONS}$$

$$\frac{\begin{array}{c} z \# \Gamma \\ \mathbf{union}\, tid = \forall\, \beta. \{\, \overline{\dot{ctor}_i : \tau_i}^{\,i} \,\} \in \Theta \\ \Theta; B; \Gamma \vdash v \leq \tau[b/\beta] \end{array}}{\Theta; B; \Gamma \vdash \dot{ctor}_j\ tid[b] v \Rightarrow \{z : tid | z = \dot{ctor}_j\ tid[b] v\}} \text{ INFER\_V\_ANF\_DATA\_CONS\_POLY}$$

$\boxed{\Theta; B; \Gamma \vdash v \leq \tau}$ Check that type of v is τ

$$\frac{\begin{array}{c} \Theta; B; \Gamma \vdash v \Rightarrow \{z_2 : b | \phi_2\} \\ \Theta; B; \Gamma \vdash \{z_2 : b | \phi_2\} \precapprox \{z_1 : b | \phi_1\} \end{array}}{\Theta; B; \Gamma \vdash v \leq \{z_1 : b | \phi_1\}} \text{ CHECK\_V\_ANF\_VAL}$$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash e \Rightarrow \tau}$ Infer that type of e is τ

$$\frac{\begin{array}{c} z_3 \# \Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v_1 \Rightarrow \{z_1 : \mathbf{int} | \phi_1\} \\ \Theta; B; \Gamma \vdash v_2 \Rightarrow \{z_2 : \mathbf{int} | \phi_2\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash v_1 + v_2 \Rightarrow \{z_3 : \mathbf{int} | z_3 = v_1 + v_2\}} \text{ INFER\_E\_ANF\_PLUS}$$

$$\frac{\begin{array}{l} z_3\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v_1 \Rightarrow \{z_1 : \mathbf{int} | \phi_1\} \\ \Theta; B; \Gamma \vdash v_2 \Rightarrow \{z_2 : \mathbf{int} | \phi_2\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash v_1 \leq v_2 \Rightarrow \{z_3 : \mathbf{bool} | z_3 = va1 \leq va2\}} \text{ INFER\_E\_ANF\_LEQ}$$

$$\frac{\begin{array}{l} \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \mathbf{val}\, f : (x : b[\phi]) \to \tau \in \Phi \\ \Theta; B; \Gamma \vdash v \leq \{z : b | \phi\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash f\, v \Rightarrow \tau[v/x]} \text{ INFER\_E\_ANF\_APP}$$

$$\frac{\begin{array}{l} \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau \in \Phi \\ \Theta; B; \Gamma \vdash v \leq \{z : b[b_2/\beta] | \phi\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash f[b_2] v \Rightarrow \tau[b_2/\beta][v/x]} \text{ INFER\_E\_ANF\_APP\_POLY}$$

$$\frac{\begin{array}{l} z\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v \Rightarrow \{z : b_1 * b_2 | \phi\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{fst}\, v \Rightarrow \{z : b_1 | z = \mathbf{fst}\, v\}} \text{ INFER\_E\_ANF\_FST}$$

$$\frac{\begin{array}{l} z\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v \Rightarrow \{z : b_1 * b_2 | \phi\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{snd}\, v \Rightarrow \{z : b_2 | z = \mathbf{snd}\, v\}} \text{ INFER\_E\_ANF\_SND}$$

$$\frac{\begin{array}{l} z\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v_1 \Rightarrow \{z_1 : \mathbf{bvec} | \phi_1\} \\ \Theta; B; \Gamma \vdash v_2 \Rightarrow \{z_2 : \mathbf{bvec} | \phi_2\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash v_1 @ v_2 \Rightarrow \{z : \mathbf{bvec} | z = v_1 @ v_2\}} \text{ INFER\_E\_ANF\_CONCAT}$$

$$\frac{\begin{array}{l} z\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v_1 \Rightarrow \{z_1 : \mathbf{int} | \phi_1\} \\ \Theta; B; \Gamma \vdash v_2 \Rightarrow \{z_2 : \mathbf{bvec} | \phi_2\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{split}\, v_1\, v_2 \Rightarrow \{z : \mathbf{bvec} | v_2 = \mathbf{fst}\, z @ \mathbf{snd}\, z \wedge v_1 = \mathbf{len}\, (\mathbf{fst}\, z)\}} \text{ INFER\_E\_ANF\_SPLIT}$$

$$\frac{\begin{array}{l} z\#\Gamma \\ \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ \Theta; B; \Gamma \vdash v \Rightarrow \{z : \mathbf{bvec} | \phi\} \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{snd}\, v \Rightarrow \{z : b_2 | z = \mathbf{len}\, v\}} \text{ INFER\_E\_ANF\_LEN}$$

$$\frac{\begin{array}{l} \Theta \vdash_{wf} \Phi \\ \Theta; B; \Gamma \vdash_{wf} \Delta \\ u : \tau \in \Delta \end{array}}{\Theta; \Phi; B; \Gamma; \Delta \vdash u \Rightarrow \tau} \text{ INFER\_E\_ANF\_MVAR}$$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash e \leq \tau}$ Check that type of e is τ

$$\frac{\Theta; \Phi; B; \Gamma; \Delta \vdash e \Rightarrow \{z_2 : b | \phi_2\} \quad \Theta; B; \Gamma \vdash \{z_2 : b|\phi_2\} \lessapprox \{z_1 : b|\phi_1\}}{\Theta; \Phi; B; \Gamma; \Delta \vdash e \leq \{z_1 : b|\phi_1\}} \text{ CHECK\_E\_ANF\_EXPR}$$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash s \leq \tau}$ Check that type of s is τ

$$\frac{\Theta \vdash_{wf} \Phi \quad \Theta; B; \Gamma \vdash_{wf} \Delta \quad \Theta; B; \Gamma \vdash v \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash v \leq \tau} \text{ CHECK\_S\_VAL}$$

$$\frac{u \notin \mathrm{dom}(\Delta) \quad \Theta; B; \Gamma \vdash v \leq \tau \quad \Theta; \Phi; B; \Gamma; \Delta, u : \tau \vdash s \leq \tau_2}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{var}\, u : \tau := v\, \mathbf{in}\, s \leq \tau_2} \text{ CHECK\_S\_VAR}$$

$$\frac{\Theta \vdash_{wf} \Phi \quad \Theta; B; \Gamma \vdash_{wf} \Delta \quad u : \tau \in \Delta \quad \Theta; B; \Gamma \vdash v \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash u := v \leq \{z : \mathbf{unit} | \top\}} \text{ CHECK\_S\_ASSIGN}$$

$$\frac{\Theta; B; \Gamma \vdash v \Rightarrow \{z : \mathbf{bool}|\phi_1\} \quad \Theta; \Phi; B; \Gamma; \Delta \vdash s_1 \leq \{z_1 : b | v = \mathbf{T} \implies \phi[z_1/z]\} \quad \Theta; \Phi; B; \Gamma; \Delta \vdash s_2 \leq \{z_2 : b | v = \mathbf{F} \implies \phi[z_2/z]\}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{if}\, v\, \mathbf{then}\, s_1\, \mathbf{else}\, s_2 \leq \{z : b | \phi\}} \text{ CHECK\_S\_IF}$$

$$\frac{x \# \Gamma \quad \Theta; \Phi; B; \Gamma; \Delta \vdash e \Rightarrow \{z : b|\phi\} \quad \Theta; \Phi; B; \Gamma, x : b[\phi[x/z]]; \Delta \vdash s \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{let}\, x = e\, \mathbf{in}\, s \leq \tau} \text{ CHECK\_S\_LET}$$

$$\frac{x \# \Gamma \quad \Theta; \Phi; B; \Gamma, x : \mathbf{bool}[\phi]; \Delta \vdash s \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{assert}\, \phi\, \mathbf{in}\, s \leq \tau} \text{ CHECK\_S\_ASSERT}$$

$$\frac{x \# \Gamma \quad \Theta; \Phi; B; \Gamma; \Delta \vdash s_1 \leq \{z : b|\phi\} \quad \Theta; \Phi; B; \Gamma, x : b[\phi[x/z]]; \Delta \vdash s_2 \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{let}\ x : \{z : b|\phi\} = s_1\ \mathbf{in}\ s_2 \leq \tau} \text{ CHECK\_S\_LET2}$$

$$\frac{\mathbf{union}\, tid = \{\, \overline{\dot{ctor}_i : \{z_i : b_i | \phi_i\}}^{\,i}\, \} \in \Theta \quad \Theta; B; \Gamma \vdash v \Rightarrow \{z : tid|\phi\} \quad \overline{\Theta; \Phi; B; \Gamma, x_i : b_i[v = \dot{ctor}_i\, tid\, x_i \wedge \phi_i[x_i/z_i]]; \Delta \vdash s_i \leq \tau}^{\,i}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{match}\, v\, \mathbf{of}\, \overline{\dot{ctor}_i\, x_i \Rightarrow s_i}^{\,i} \leq \tau} \text{ CHECK\_S\_MATCH}$$

$$\frac{\Theta; \Phi; B; \Gamma; \Delta \vdash s_1 \leq \{z : \mathbf{bool}|\top\} \quad \Theta; \Phi; B; \Gamma; \Delta \vdash s_2 \leq \{z : \mathbf{unit}|\top\}}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{while}\, (s_1)\, \mathbf{do}\, \{s_2\} \leq \{z : \mathbf{unit}|\top\}} \text{ CHECK\_S\_WHILE}$$

$$\frac{\Theta; \Phi; B; \Gamma; \Delta \vdash s_1 \leq \{z : \mathbf{unit}|\top\} \quad \Theta; \Phi; B; \Gamma; \Delta \vdash s_2 \leq \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash s_1; s_2 \leq \tau} \text{ CHECK\_S\_SEQ}$$

$$\frac{}{\Theta; \Phi; B; \Gamma; \Delta \vdash \mathbf{abort} \leq \tau} \quad \text{CHECK\_S\_ABORT}$$

$\boxed{\Theta_1; \Phi_1 \vdash def_1 \,..\, def_n \rightsquigarrow \Theta_2; \Phi_2}$

$$\frac{\begin{array}{c}\mathbf{val}\, f : (x : b[\phi]) \to \tau \in \Phi \\ \Theta; \Phi; \cdot; x : b[\phi]; \cdot \vdash s \leq \tau\end{array}}{\Theta; \Phi \vdash \mathbf{function}\, f(x) = s \rightsquigarrow \Theta; \Phi, \mathbf{function}\, f(x) = s} \quad \text{CHECK\_DEFS\_ANF\_FUNDEF}$$

$$\frac{\begin{array}{c}\mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau \in \Phi \\ \Theta; \Phi; \cdot, \beta; x : b[\phi]; \cdot \vdash s \leq \tau\end{array}}{\Theta; \Phi \vdash \mathbf{function}\, f(x) = s \rightsquigarrow \Theta; \Phi, \mathbf{function}\, f(x) = s} \quad \text{CHECK\_DEFS\_ANF\_FUNDEF\_POLY}$$

$$\frac{\Theta \vdash_{wf} \mathbf{val}\, f : (x : b[\phi]) \to \tau}{\Theta; \Phi \vdash \mathbf{val}\, f : (x : b[\phi]) \to \tau \rightsquigarrow \Theta; \Phi, \mathbf{val}\, f : (x : b[\phi]) \to \tau} \quad \text{CHECK\_DEFS\_ANF\_VALSPEC}$$

$$\frac{\Theta \vdash_{wf} \mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau}{\Theta; \Phi \vdash \mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau \rightsquigarrow \Theta; \Phi, \mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau} \quad \text{CHECK\_DEFS\_ANF\_VALSPEC\_POLY}$$

$$\frac{}{\Theta; \Phi \vdash \mathbf{union}\, tid = \{\, \overline{ctor_i : \tau_i}^{\,i} \,\} \rightsquigarrow \Theta, \mathbf{union}\, tid = \{\, \overline{ctor_i : \tau_i}^{\,i} \,\}; \Phi} \quad \text{CHECK\_DEFS\_ANF\_UNIONDEF}$$

$$\frac{\begin{array}{c}\Theta_1; \Phi_1 \vdash def \rightsquigarrow \Theta_2; \Phi_2 \\ \Theta_2; \Phi_2 \vdash def_1 \,..\, def_n \rightsquigarrow \Theta_3; \Phi_3\end{array}}{\Theta_1; \Phi_1 \vdash def\, def_1 \,..\, def_n \rightsquigarrow \Theta_3; \Phi_3} \quad \text{CHECK\_DEFS\_ANF\_DEFS}$$

$\boxed{\vdash p}$

$$\frac{\begin{array}{c}\cdot; \cdot \vdash def_1 \,..\, def_n \rightsquigarrow \Theta_2; \Phi_2 \\ \Theta_2; \Phi_2; \cdot; \cdot; \cdot \vdash s \leq \{z : \mathbf{int} | \top\}\end{array}}{\vdash def_1; \,..\,; def_n; ; s} \quad \text{CHECK\_PROGRAM\_PROG}$$

$\boxed{\Theta \vdash \Delta \sim \delta}$

$$\frac{\begin{array}{l}\delta = u_1 \to v_1, \,..\, , u_n \to v_n \\ \Delta = u_1 : \tau_1, \,..\, , u_n : \tau_n \\ \Theta; \cdot; \cdot \vdash v_1 \leq \tau_1 \quad .. \quad \Theta; \cdot; \cdot \vdash v_n \leq \tau_n\end{array}}{\Theta \vdash \Delta \sim \delta} \quad \text{DSIM\_DSIM}$$

$\boxed{\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau}$ Program state typing judgement

$$\frac{\begin{array}{c}\Theta \vdash \Delta \sim \delta \\ \Theta; \Phi; \cdot; \cdot; \Delta \vdash s \leq \tau\end{array}}{\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau} \quad \text{CHECK\_REDEX\_STMT}$$

2.5 Operational semantics

$\boxed{\Phi \vdash \langle \delta, s_1 \rangle \to \langle \delta', s_2 \rangle}$ One step reduction

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{if}\, \mathbf{T}\, \mathbf{then}\, s_1\, \mathbf{else}\, s_2 \rangle \to \langle \delta, s_1 \rangle} \quad \text{REDUCE\_IF\_TRUE}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{if}\, \mathbf{F}\, \mathbf{then}\, s_1\, \mathbf{else}\, s_2 \rangle \to \langle \delta, s_2 \rangle} \quad \text{REDUCE\_IF\_FALSE}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{let}\, x = v\, \mathbf{in}\, s \rangle \to \langle \delta, s[v/x] \rangle} \quad \text{REDUCE\_LET\_VALUE}$$

$$\frac{v_1 + v_2 = v}{\Phi \vdash \langle \delta, \mathbf{let}\, x = v_1 + v_2\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_PLUS}$$

$$\frac{v_1 \leq v_2 = v}{\Phi \vdash \langle \delta, \mathbf{let}\, x = v_1 \leq v_2\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_LEQ}$$

$$\frac{\begin{array}{c}\mathbf{val}\, f : (x : b[\phi]) \to \tau \in \Phi \\ \mathbf{function}\, f(x) = s_1 \in \Phi\end{array}}{\Phi \vdash \langle \delta, \mathbf{let}\, y = f\, v\, \mathbf{in}\, s_2\rangle \to \langle \delta, \mathbf{let}\ y : \tau[v/x] = s_1[v/x]\ \mathbf{in}\ s_2\rangle} \quad \text{REDUCE\_LET\_APP}$$

$$\frac{\begin{array}{c}\mathbf{val} \forall\, \beta. f : (x : b[\phi]) \to \tau \in \Phi \\ \mathbf{function}\, f(x) = s_1 \in \Phi\end{array}}{\Phi \vdash \langle \delta, \mathbf{let}\, y = f[b_1] v\, \mathbf{in}\, s_2\rangle \to \langle \delta, \mathbf{let}\ y : \tau[v/x][b_1/\beta] = s_1[v/x][b_1/\beta]\ \mathbf{in}\ s_2\rangle} \quad \text{REDUCE\_LET\_APP\_POLY}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{let}\, x = \mathbf{fst}\, (v_1, v_2)\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v_1\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_FST}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{let}\, x = \mathbf{snd}\, (v_1, v_2)\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v_2\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_SND}$$

$$\frac{v_1 @ v_2 = v_3}{\Phi \vdash \langle \delta, \mathbf{let}\, x = v_1 @ v_2\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v_3\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_CONCAT}$$

$$\frac{v_1 = \mathbf{split}\, v_2\, v_3}{\Phi \vdash \langle \delta, \mathbf{let}\, x = \mathbf{split}\, v_2\, v_3\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v_1\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_SPLIT}$$

$$\frac{\mathbf{len}\, v_1 = v_2}{\Phi \vdash \langle \delta, \mathbf{let}\, x = \mathbf{len}\, v_1\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v_2\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_LEN}$$

$$\frac{v = \delta(u)}{\Phi \vdash \langle \delta, \mathbf{let}\, x = u\, \mathbf{in}\, s\rangle \to \langle \delta, \mathbf{let}\, x = v\, \mathbf{in}\, s\rangle} \quad \text{REDUCE\_LET\_MVAR}$$

$$\frac{u \notin \mathrm{dom}(\delta)}{\Phi \vdash \langle \delta, \mathbf{var}\, u : \tau := v\, \mathbf{in}\, s\rangle \to \langle \delta[u \mapsto v], s\rangle} \quad \text{REDUCE\_MVAR\_DECL}$$

$$\frac{\delta' = \delta[u \mapsto v]}{\Phi \vdash \langle \delta, u := v\rangle \to \langle \delta', ()\rangle} \quad \text{REDUCE\_MVAR\_ASSIGN}$$

$$\frac{\Phi \vdash \langle \delta, s_1\rangle \to \langle \delta', s_3\rangle}{\Phi \vdash \langle \delta, s_1; s\rangle \to \langle \delta', s_3; s\rangle} \quad \text{REDUCE\_SEQ1}$$

$$\frac{}{\Phi \vdash \langle \delta, (); s\rangle \to \langle \delta, s\rangle} \quad \text{REDUCE\_SEQ2}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{let}\ x : \tau = v\ \mathbf{in}\ s_2\rangle \to \langle \delta, s_2[v/x]\rangle} \quad \text{REDUCE\_LET2\_VAL}$$

$$\frac{\Phi \vdash \langle \delta, s_1\rangle \to \langle \delta', s_3\rangle}{\Phi \vdash \langle \delta, \mathbf{let}\ x : \tau = s_1\ \mathbf{in}\ s_2\rangle \to \langle \delta', \mathbf{let}\ x : \tau = s_3\ \mathbf{in}\ s_2\rangle} \quad \text{REDUCE\_LET2\_STMT}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{match}\, (\dot{ctor}_j\ tid\ v)\ \mathbf{of}\ \overline{\dot{ctor}_i\, x_i \Rightarrow s_i}^{\,i}\rangle \to \langle \delta, s_j[v/x_j]\rangle} \quad \text{REDUCE\_MATCH}$$

$$\frac{x\, \mathbf{fresh}}{\Phi \vdash \langle \delta, \mathbf{while}\, (s_1)\, \mathbf{do}\, \{s_2\}\rangle \to \langle \delta, \mathbf{let}\ x : \{z : \mathbf{bool} | \top\} = s_1\ \mathbf{in}\ \mathbf{if}\, x\, \mathbf{then}\, (s_2; \mathbf{while}\, (s_1)\, \mathbf{do}\, \{s_2\})\, \mathbf{else}\, ()\rangle} \quad \text{REDUCE}$$

$$\frac{}{\Phi \vdash \langle \delta, \mathbf{assert}\, \phi\, \mathbf{in}\, v\rangle \to \langle \delta, v\rangle} \quad \text{REDUCE\_ASSERT1}$$

$$\frac{\Phi \vdash \langle \delta, s_1\rangle \to \langle \delta', s_2\rangle}{\Phi \vdash \langle \delta, \mathbf{assert}\, \phi\, \mathbf{in}\, s_1\rangle \to \langle \delta', \mathbf{assert}\, \phi\, \mathbf{in}\, s_2\rangle} \quad \text{REDUCE\_ASSERT2}$$

$\boxed{\Phi \vdash \langle \delta_1, s_1 \rangle \xrightarrow{*} \langle \delta_2, s_2 \rangle}$ Multi-step reduction

$$\frac{\Phi \vdash \langle \delta_1, s_1 \rangle \to \langle \delta_2, s_2 \rangle}{\Phi \vdash \langle \delta_1, s_1 \rangle \xrightarrow{*} \langle \delta_2, s_2 \rangle} \text{ REDUCE\_MANY\_SINGLE\_STEP}$$

$$\frac{\Phi \vdash \langle \delta_1, s_1 \rangle \to \langle \delta_2, s_2 \rangle \quad \Phi \vdash \langle \delta_2, s_2 \rangle \xrightarrow{*} \langle \delta_3, s_3 \rangle}{\Phi \vdash \langle \delta_1, s_1 \rangle \xrightarrow{*} \langle \delta_3, s_3 \rangle} \text{ REDUCE\_MANY\_MANY\_STEP}$$

2.6 Machine configuration check

$\boxed{\Theta \vdash \delta \sim \Delta}$

$$\frac{}{\Theta \vdash \cdot \sim \cdot} \text{ CHECK\_STORE\_EMPTY}$$

$$\frac{u \notin \mathrm{dom}(\Delta) \quad \Theta \vdash \delta \sim \Delta \quad \Theta; \cdot; \cdot \vdash v \leq \tau}{\Theta \vdash \delta[u \mapsto v] \sim \Delta, u : \tau} \text{ CHECK\_STORE\_CONS}$$

$\boxed{\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau}$

$$\frac{\Theta \vdash \delta \sim \Delta \quad \Theta; \Phi; \cdot; \cdot; \Delta \vdash s \leq \tau}{\Theta; \Phi; \Delta \vdash (\delta, s) \leq \tau} \text{ CHECK\_CONFIG\_CONFIG}$$

$\boxed{E \vdash \textbf{pack\_record}\, \tau\, x\, id_1 = x_1 \,...\, id_n = x_n \rightsquigarrow L}$

$\boxed{E \vdash \textbf{unpack\_field}\, \tau\, x\, x'\, id \rightsquigarrow L}$

$\boxed{E \vdash \textbf{update\_record}\, \tau\, x\, x'\, id_1 = x_1 \,...\, id_n = x_n \rightsquigarrow L}$

$\boxed{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} L : \gamma}$ WF for let-context

$$\frac{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} e : b}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \textbf{let}\, x = e\, \textbf{in}\, \_ : x : \{z : b | \phi\}} \text{ WF\_LCTX\_LET}$$

$$\frac{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} s : b \quad \Theta; B; \Gamma \vdash_{wf} \tau}{\Theta; \Phi; B; \Gamma; \Delta \vdash_{wf} \textbf{let}\ x : \tau = s\ \textbf{in}\ \_ : x : \{z : b | \phi\}} \text{ WF\_LCTX\_LET2}$$

3 Sail to MiniSail-ANF conversion

3.1 Converting types

$\boxed{typquant \rightsquigarrow kinded\_id_1 \,..\, kinded\_id_m, n\_constraint}$

Normalise typequant. Pull out all of the constraints and put them at the end $\boxed{E \vdash typ \rightsquigarrow \tau}$

Convert Sail type to MiniSail type. First form is that we normalise bringing out any exisentials to the top level.

$$\frac{E; \epsilon \vdash typ; z \rightsquigarrow b; \phi}{E \vdash typ \rightsquigarrow \{z : b | \phi\}} \text{ TYP\_CONV}$$

$\boxed{E; M \vdash typ\_arg \rightsquigarrow \phi}$

$\boxed{E; M \vdash typ\_arg \rightsquigarrow ce}$

$\boxed{E; M \vdash typ; ce \rightsquigarrow b; \phi}$

Extract MiniSail base type and constraint from Sail type.

$$\frac{}{E; M \vdash \mathbf{int}; ce \rightsquigarrow \mathbf{int}; \top} \text{ CTA\_INT}$$

$$\frac{E; M \vdash typ\_arg \rightsquigarrow ce'}{E; M \vdash \mathbf{atom}(typ\_arg); ce \rightsquigarrow \mathbf{int}; ce = ce'} \text{ CTA\_ATOM\_INT}$$

$$\frac{}{E; M \vdash \mathbf{bool}; ce \rightsquigarrow \mathbf{bool}; \top} \text{ CTA\_BOOL}$$

$$\frac{E; M \vdash typ\_arg \rightsquigarrow \phi}{E; M \vdash \mathbf{atom\_bool}(typ\_arg); ce \rightsquigarrow \mathbf{bool}; \phi} \text{ CTA\_ATOM\_BOOL}$$

$$\frac{\begin{array}{c} E; M \vdash typ\_arg_1 \rightsquigarrow ce_1 \\ E; M \vdash typ\_arg_2 \rightsquigarrow ce_2 \end{array}}{E; M \vdash \mathbf{range}(typ\_arg_1, typ\_arg_2); ce \rightsquigarrow \mathbf{int}; ce_1 \leq ce \wedge ce \leq ce_2} \text{ CTA\_RANGE}$$

$$\frac{\begin{array}{c} M' = M, ce, kinded\_id_1 \,..\, kinded\_id_m \\ E; M' \vdash typ; ce \rightsquigarrow b; \phi \\ E; M' \vdash n\_constraint \rightsquigarrow \phi' \end{array}}{E; M \vdash \{kinded\_id_1 \,..\, kinded\_id_m, n\_constraint.typ\}; ce \rightsquigarrow b; \phi \wedge \phi'} \text{ CTA\_EXIST}$$

$$\frac{\begin{array}{c} E; M \vdash typ; \mathbf{fst}\, ce \rightsquigarrow b; \phi \\ E; M \vdash (typ_1, \,..\, , typ_n); \mathbf{snd}\, ce \rightsquigarrow b'; \phi' \end{array}}{E; M \vdash (typ, typ_1, \,..\, , typ_n); ce \rightsquigarrow b * b'; \phi \wedge \phi'} \text{ CTA\_TUPLE}$$

$\boxed{E; M \vdash n\_constraint \rightsquigarrow \phi}$

Convert Sail constraint to MiniSail constraint.

$$\frac{\begin{array}{c} E; M \vdash nexp_1 \rightsquigarrow ce_1 \\ E; M \vdash nexp_2 \rightsquigarrow ce_2 \end{array}}{E; M \vdash nexp_1 \equiv nexp_2 \rightsquigarrow ce_1 = ce_2} \text{ CONVERT\_C\_EQUAL}$$

$\boxed{E; M \vdash nexp \rightsquigarrow ce}$

Convert Sail constraint expression to MiniSail constraint expression.

$$\frac{ce = M(kid)}{E; M \vdash kid \rightsquigarrow ce} \text{ NEXP\_CEA\_VAR}$$

$$\frac{\begin{array}{c} E; M \vdash nexp_1 \rightsquigarrow ce_1 \\ E; M \vdash nexp_2 \rightsquigarrow ce_2 \end{array}}{E; M \vdash nexp_2 + nexp_1 \rightsquigarrow ce_1 + ce_2} \text{ NEXP\_CEA\_ADD}$$

3.2 Converting expressions

$\boxed{lit \rightsquigarrow lp}$

$\boxed{lit \rightsquigarrow l}$

$$\frac{}{num \rightsquigarrow n}\ \text{CL\_NUM}$$

$\boxed{E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta}$

$\boxed{E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; L : \tau}$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ lit \rightsquigarrow l \\ E \vdash typ \rightsquigarrow \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta\end{array}}{E \vdash lit : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; \textbf{let}\, x = l\, \textbf{in}\, \_ : \tau}\ \text{CE\_LIT}$$

$$\frac{\begin{array}{l}id/\textbf{immutable} : id \in E \\ id \sim x \\ E \vdash typ \rightsquigarrow \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta\end{array}}{E \vdash id : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/x : \tau \vdash x; \_ : \tau}\ \text{CE\_IMMUTABLE}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ id/\textbf{enum} : typ \in E \\ E \vdash id \rightsquigarrow \dot{ctor}, tid \\ E \vdash typ \rightsquigarrow \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta\end{array}}{E \vdash id : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/x : \tau \vdash x; \textbf{let}\, x = \dot{ctor}\, tid\, ()\, \textbf{in}\, \_ : \tau}\ \text{CE\_ENUM}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ id/\textbf{mutable} : typ \in E \\ id \sim u \\ E \vdash typ \rightsquigarrow \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta\end{array}}{E \vdash id : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/x : \tau \vdash x; \textbf{let}\, x = u\, \textbf{in}\, \_ : \tau}\ \text{CE\_MUTABLE}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ id/\textbf{register} : typ \in E \\ id \sim u \\ E \vdash typ \rightsquigarrow \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta\end{array}}{E \vdash id : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/x : \tau \vdash x; \textbf{let}\, x = u\, \textbf{in}\, \_ : \tau}\ \text{CE\_REGISTER}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp} \vdash exp : \mathrm{typ}_{exp} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E_2 \vdash (exp_1, .., exp_n) : typ \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2/\gamma_2 \vdash x_2; L_2 : \tau_2 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash (exp, exp_1, .., exp_n) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1', \gamma_2', x : \tau \vdash x; L_1[L_2[\textbf{let}\, x = (x', x'')\, \textbf{in}\, \_]] : \tau}\ \text{CE\_TUPLE}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{(exp_1,..,exp_n)} \vdash (exp_1,\, ..\,, exp_n) : \mathrm{typ}_{(exp_1,..,exp_n)} \rightsquigarrow \Theta'; \Phi'; B'; \Gamma'; \Delta'/\gamma' \vdash x'; L : \tau' \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash \textbf{inst\_of}\ id(exp_1,\, ..\,, exp_n) \rightsquigarrow x''; L''\end{array}}{E \vdash id(exp_1,\, ..\,, exp_n) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma', x : \tau \vdash x; L''[L[\textbf{let}\, x = f\,(x'', x')\, \textbf{in}\, \_]] : \tau}\ \text{CE\_APP}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{(exp_1,..,exp_n)} \vdash (exp_1,\, ..\,, exp_n) : \mathrm{typ}_{(exp_1,..,exp_n)} \rightsquigarrow \Theta'; \Phi'; B'; \Gamma'; \Delta'/\gamma' \vdash x'; L : \tau' \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash id \rightsquigarrow \dot{ctor}, tid \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash id(exp_1,\, ..\,, exp_n) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma', x : \tau \vdash x; L[\textbf{let}\, x = \dot{ctor}\ tid\ x'\, \textbf{in}\, \_] : \tau}\ \text{CE\_CTOR}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2/\gamma_2 \vdash x_2; L_2 : \tau_2 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash exp_1\ +\ exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, \gamma_2, x : \tau \vdash x; L_1[L_2[\textbf{let}\, x = x_1 + x_2\, \textbf{in}\, \_]] : \tau}\ \text{CE\_PLUS}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2/\gamma_2 \vdash x_2; L_2 : \tau_2 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash exp_1\ \leq\ exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, \gamma_2, x : \tau \vdash x; L_1[L_2[\textbf{let}\, x = x_1 \leq x_2\, \textbf{in}\, \_]] : \tau_1}\ \text{CE\_LEQ}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash \textbf{len}(exp_1) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, x : \tau \vdash x; L_1[\textbf{let}\, x = \textbf{len}\, x_1\, \textbf{in}\, \_] : \tau}\ \text{CE\_LEN}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_2; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2/\gamma_2 \vdash x_2; L_2 : \tau_2 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash exp_1 @ exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, \gamma_2, x : \tau \vdash x; L_1[L_2[\textbf{let}\, x = x_1 @ x_2\, \textbf{in}\, \_]] : \tau}\ \text{CE\_CONCAT}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash \textbf{fst}(exp_1) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, x : \tau \vdash x; L_1[\textbf{let}\, x = \textbf{fst}\, x_1\, \textbf{in}\, \_] : \tau_1}\ \text{CE\_FST}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau\end{array}}{E \vdash \textbf{snd}(exp_1) : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1, x : \tau \vdash x; L_1[\textbf{let}\, x = \textbf{snd}\, x_1\, \textbf{in}\, \_] : \tau}\ \text{CE\_SND}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ \overline{E \vdash exp_i : typ_i \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x_i; L_i : \tau_i}^{\,i\in 1\ldots n} \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash \textbf{pack\_record}\, \tau\, x\, \overline{id_i = x_i}^{\,i\in 1\ldots n} \rightsquigarrow L\end{array}}{E \vdash \textbf{struct}\, \{\, \overline{id_i = exp_i}^{\,i\in 1\ldots n} \,\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; (L_1 + \ldots + L_n)[L] : \tau}\ \text{CE\_RECORD}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E \vdash exp : typ' \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x'; L : \tau' \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash \textbf{unpack\_field}\, \tau\, x\, x'\, id \rightsquigarrow L'\end{array}}{E \vdash exp.id : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; L[L'] : \tau}\ \text{CE\_FIELD}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E \vdash exp : typ' \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x'; L : \tau' \\ \overline{E \vdash exp_i : typ_i \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x_i; L_i : \tau_i}^{\,i\in 0\ldots n} \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash \textbf{update\_record}\, \tau\, x\, x'\, \overline{id_i = x_i}^{\,i\in 0\ldots n} \rightsquigarrow L'\end{array}}{E \vdash \{exp\, \textbf{with}\, \overline{id_i = exp_i}^{\,i\in 0\ldots n}\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; (L_0 + \ldots + L_n)[L'] : \tau}\ \text{CE\_RECORD\_UPDATE}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E \vdash \textbf{if}\, exp_1\, \textbf{then}\, exp_2\, \textbf{else}\, exp_3 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau\end{array}}{E \vdash \textbf{if}\, exp_1\, \textbf{then}\, exp_2\, \textbf{else}\, exp_3 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\epsilon \vdash x; \textbf{let}\, x : \tau = s\, \textbf{in}\, \_ : \tau}\ \text{CE\_IF}$$

$$\frac{\begin{array}{l}\textbf{fresh}\, x \\ E \vdash \textbf{match}\, exp\{pat_1 \rightarrow exp_1, \ldots, pat_n \rightarrow exp_n\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau\end{array}}{E \vdash \textbf{match}\, exp\{pat_1 \rightarrow exp_1, \ldots, pat_n \rightarrow exp_n\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\epsilon \vdash x; \textbf{let}\, x : \tau = s\, \textbf{in}\, \_ : \tau}\ \text{CE\_MATCH}$$

$\boxed{E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}$

$$\frac{\begin{array}{l}id \sim x \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L : \tau_1 \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash (pat \Rightarrow exp_2) : |\tau_1|_b/x \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s_2 : \tau'\end{array}}{E \vdash \textbf{let}\, pat = exp_1\, \textbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[s_2] : \tau}\ \text{CS\_LET}$$

$$\frac{\begin{array}{l}id \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L : \tau' \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ id/\textbf{mutable} \notin E\end{array}}{E \vdash \textbf{var}\, id = exp_1\, \textbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[\textbf{var}\, u : \tau := x_1\, \textbf{in}\, s_2] : \tau}\ \text{CS\_VAR}$$

$$\frac{\begin{array}{l}id \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ E \vdash typ' \rightsquigarrow \tau'\end{array}}{E \vdash \textbf{var}\, (typ')id = exp_1\, \textbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[\textbf{var}\, u : \tau' := x_1\, \textbf{in}\, s_2] : \tau}\ \text{CS\_CAST}$$

$$\frac{\begin{array}{l} id \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L : \tau' \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ id/\mathbf{mutable} : typ' \in E \end{array}}{E \vdash \mathbf{var}\, id = exp_1\, \mathbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[u := x_1; s_2] : \tau} \text{ CS\_ASSIGN}$$

$$\frac{\begin{array}{l} id_1 \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_3; L[\mathbf{let}\, x_2 = u\, \mathbf{in}\, \_\_] : \tau' \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash \mathbf{update\_record}\, \tau\, x_1\, x_2\, id_2 = x_3 \rightsquigarrow L' \end{array}}{E \vdash \mathbf{var}\, id_1.id_2 = exp_1\, \mathbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[L'[u := x_1; s_2]] : \tau} \text{ CS\_FIELD\_ASSIGN}$$

$$\frac{\begin{array}{l} id \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L : \tau' \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ id/\mathbf{register} : typ' \in E \end{array}}{E \vdash \mathbf{var}\, \mathbf{deref}\, id = exp_1\, \mathbf{in}\, exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[u := x_1; s_2] : \tau} \text{ CS\_DEREF}$$

$$\frac{\begin{array}{l} id \sim u \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x_1; L_1 : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2/\gamma_2 \vdash x_2; L_2 : \tau_2 \\ E_{exp_3} \vdash exp_3 : \mathrm{typ}_{exp_3} \rightsquigarrow \Theta_3; \Phi_3; B_3; \Gamma_3; \Delta_3/\gamma_3 \vdash x_3; L_3 : \tau_3 \\ E_{exp_4} \vdash exp_4 : \mathrm{typ}_{exp_4} \rightsquigarrow \Theta_4; \Phi_4; B_4; \Gamma_4; \Delta_4 \vdash s_4 : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \\ L_4 = \mathbf{let}\, x = u\, \mathbf{in}\, \_\_ \\ L_5 = \mathbf{let}\, x_4 = \mathbf{update\_vector\_range}\, x\, x_1\, x_2\, x_3\, \mathbf{in}\, \_\_ \end{array}}{E \vdash \mathbf{var}\, id[exp_1..exp_2] = exp_3\, \mathbf{in}\, exp_4 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash (L_1 + L_2 + L_3 + L_4 + L_5)[u := x_4; s_4] : \tau} \text{ CS\_VECT}$$

$$\frac{\begin{array}{l} E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \end{array}}{E \vdash \{exp\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau} \text{ CS\_BLOCK\_SINGLE}$$

$$\frac{\begin{array}{l} E \vdash exp : \mathrm{typ}_{exp} \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau \\ E \vdash \{exp_1; \ldots; exp_n\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s' : \tau' \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \end{array}}{E \vdash \{exp; exp_1; \ldots; exp_n\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s; s' : \tau'} \text{ CS\_BLOCK\_CONS}$$

$$\frac{\begin{array}{l} E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1/\gamma_1 \vdash x; L : \tau' \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau_2 \\ E_{exp_3} \vdash exp_2 : \mathrm{typ}_{exp_3} \rightsquigarrow \Theta_3; \Phi_3; B_3; \Gamma_3; \Delta_3 \vdash s_3 : \tau_3 \\ E \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \\ E \vdash typ \rightsquigarrow \tau \end{array}}{E \vdash \mathbf{if}\, exp_1\, \mathbf{then}\, exp_2\, \mathbf{else}\, exp_3 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[\mathbf{if}\, x\, \mathbf{then}\, s_2\, \mathbf{else}\, s_3] : \tau} \text{ CS\_IF}$$

$$\frac{\begin{array}{l} E_{exp} \vdash exp : \mathrm{typ}_{exp} \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma_1 \vdash x; L : \tau' \\ E \vdash (pat_1 \Rightarrow exp_1), \ldots, (pat_n \Rightarrow exp_n) : b/x \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau' \end{array}}{E \vdash \mathbf{match}\, exp\{pat_1 \rightarrow exp_1, \ldots, pat_n \rightarrow exp_n\} : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[s] : \tau} \text{ CS\_MATCH}$$

$$\frac{\begin{array}{l}E \vdash \mathrm{typ}_{exp_1} \rightsquigarrow \{z : b | \phi\} \\ E_{exp_1} \vdash exp_1 : \mathrm{typ}_{exp_1} \rightsquigarrow \Theta_1; \Phi_1; B_1; \Gamma_1; \Delta_1 \vdash s_1 : \tau_1 \\ E_{exp_2} \vdash exp_2 : \mathrm{typ}_{exp_2} \rightsquigarrow \Theta_2; \Phi_2; B_2; \Gamma_2; \Delta_2 \vdash s_2 : \tau_2\end{array}}{E \vdash \textbf{while}\ exp_1\ exp_2 : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash \textbf{while}\,(s_1)\,\textbf{do}\,\{\textbf{assert}\,\phi\,\textbf{in}\,s_2\} : \tau}\ \text{CS\_WHILE}$$

$$\frac{E_{exp} \vdash exp : \mathrm{typ}_{exp} \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta/\gamma \vdash x; L : \tau}{E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash L[x] : \tau}\ \text{CS\_EXPR}$$

$\boxed{E \vdash \Pi : b_1/x_1 \,..\, b_n/x_n \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}$ Convert match branches

$$\frac{E \vdash exp : typ \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}{E \vdash \Rightarrow exp, \Pi : \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}\ \text{CB\_EMPTY}$$

$$\frac{E \vdash \Pi : \textbf{unit}/x \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}{E \vdash () \Rightarrow exp, \Pi : \textbf{unit}/x \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau}\ \text{CB\_UNIT}$$

$$\frac{\begin{array}{l}b \in \{\textbf{int}, \textbf{bool}\} \\ E \vdash \Pi \rightsquigarrow \Pi_1; lp_1 || \,..\, || \Pi_n; lp_n \\ \overline{E \vdash \Pi_i : b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s_i : \tau}^{\,i \in 1..n}\end{array}}{E \vdash \Pi : b/x\, b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash \textbf{switch}\, x \{\, \overline{lp_i \Rightarrow s_i}^{\,i \in 1..n}\,\} : \tau}\ \text{CB\_GROUND}$$

$$\frac{\begin{array}{l}E \vdash \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\, b'_1\, x'_1 || \,..\, || \Pi_n; \dot{ctor}_n\, b'_n\, x'_n \\ \overline{E \vdash \Pi_i : b'_1/x'_1\, b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s_i : \tau}^{\,i \in 1..n}\end{array}}{E \vdash \Pi : tid/x\, b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash \textbf{match}\, x\, \textbf{of}\, \overline{\dot{ctor}_i\, x'_i \Rightarrow s_i}^{\,i \in 1..n} : \tau}\ \text{CB\_CTOR}$$

$$\frac{\begin{array}{l}b = (b'_1, \,...\, , b'_n) \\ E \vdash \Pi : b \rightsquigarrow \Pi'; b'_1/x'_1 \,...\, b'_n/x'_n \\ E \vdash \Pi' : b'_1/x'_1 \,...\, b'_n/x'_n\, b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau\end{array}}{E \vdash \Pi : b/x\, b_1/x_1 \,..\, b_m/x_m \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash \textbf{unpack}\, x\, \textbf{into}\, x'_1, \,...\, , x'_n\, \textbf{in}\, s : \tau}\ \text{CB\_TUPLE}$$

3.3 Convert patterns

$\boxed{E \vdash x_1; x_2; pat_1, \,...\, , pat_n \rightsquigarrow L; x_3}$

$$\frac{\begin{array}{l}\textbf{fresh}\, x'_0 \,..\, x'_4 \\ lit \rightsquigarrow l \\ L_1 = \textbf{let}\, x'_0 = (\textbf{len}\, l)\, \textbf{in let}\, x'_1 = \textbf{split}\, x_1\, x'_0\, \textbf{in}\, \_ \\ L_2 = \textbf{let}\, x'_2 = (\textbf{fst}\, x'_1)\, \textbf{in let}\, x'_3 = \textbf{snd}\, x'_1\, \textbf{in let}\, x'_4 = \textbf{eqand}\, x_2\, x'_2\, l\, \textbf{in}\, \_ \\ E \vdash x'_3; x'_4; pat_1, \,..\, , pat_n \rightsquigarrow L_3; x_3\end{array}}{E \vdash x_1; x_2; lit, pat_1, \,..\, , pat_n \rightsquigarrow L_1[L_2[L_3]]; x_3}\ \text{PHV\_LIT}$$

$\boxed{E \vdash \Pi \rightsquigarrow \Pi_1; lp_1 || \,..\, || \Pi_n; lp_n}$

$$\frac{}{E \vdash \rightsquigarrow}\ \text{PHG\_EMPTY}$$

$$\frac{\begin{array}{l}lit \rightsquigarrow l \\ E \vdash \Pi \rightsquigarrow \overline{\Pi_i; lp_i}^{\,i \in 1..q} || \Pi'; l || \overline{\Pi'_i; lp'_i}^{\,i \in 1..m}\end{array}}{E \vdash (lit\, pat_1 \,..\, pat_n \Rightarrow exp), \Pi \rightsquigarrow \overline{\Pi_i; lp_i}^{\,i \in 1..q} || (pat_1 \,..\, pat_n \Rightarrow exp), \Pi'; l || \overline{\Pi'_i; lp'_i}^{\,i \in 1..m}}\ \text{PHG\_LIT1}$$

$$\frac{\begin{array}{l} lit \rightsquigarrow l \\ l \notin lp_1 .. lp_m \\ E \vdash \Pi \rightsquigarrow \Pi_1; lp_1 || .. || \Pi_m; lp_m \end{array}}{E \vdash (lit\ pat_1 .. pat_n \Rightarrow exp), \Pi \rightsquigarrow (pat_1 .. pat_n \Rightarrow exp); l || \Pi_1; lp_1 || .. || \Pi_m; lp_m}\ \text{PHG\_LIT2}$$

$$\frac{}{E \vdash (\_\ pat_1 .. pat_n \Rightarrow exp), \Pi \rightsquigarrow (pat_1 .. pat_n \Rightarrow exp); \_}\ \text{PHG\_WILD}$$

$$\frac{}{E \vdash (id\ pat_1 .. pat_n \Rightarrow exp), \Pi \rightsquigarrow (pat_1 .. pat_n \Rightarrow exp); id}\ \text{PHG\_VAR}$$

$\boxed{E \vdash \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\ b_1\ x_1 || .. || \Pi_n; \dot{ctor}_n\ b_n\ x_n}$

$$\frac{}{E \vdash \rightsquigarrow}\ \text{PHC\_EMPTY}$$

$$\frac{\begin{array}{l} E \vdash id \rightsquigarrow \dot{ctor}, tid \\ E \vdash \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\ b_1\ x_1 || .. || \Pi_n; \dot{ctor}_n\ b_n\ x_n \end{array}}{E \vdash id(pat'_1, .., pat'_m)\ pat_1 .. pat_n \Rightarrow exp, \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\ b_1\ x_1 || .. || \Pi_n; \dot{ctor}_n\ b_n\ x_n}\ \text{PHC\_CTOR}$$

$$\frac{E \vdash \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\ b_1\ x_1 || .. || \Pi_n; \dot{ctor}_n\ b_n\ x_n}{E \vdash id\ pat_1 .. pat_n \Rightarrow exp, \Pi \rightsquigarrow \Pi_1; \dot{ctor}_1\ b_1\ x_1 || .. || \Pi_n; \dot{ctor}_n\ b_n\ x_n}\ \text{PHC\_VAR}$$

$\boxed{E \vdash \Pi : b \rightsquigarrow \Pi'; b_1/x_1 .. b_n/x_n}$

$$\frac{}{E \vdash : b \rightsquigarrow ;}\ \text{PHT\_EMPTY}$$

$$\frac{\begin{array}{l} \textbf{fresh}\ x_1 .. x_n \\ b = (b_1, .., b_n) \end{array}}{E \vdash (pat_1, .., pat_n)\ pat'_1 .. pat'_m \Rightarrow exp, \Pi : b \rightsquigarrow pat_1 .. pat_n\ pat'_1 .. pat'_m \Rightarrow exp, \Pi; b_1/x_1 .. b_n/x_n}\ \text{PHT\_TUPLE}$$

$$\frac{\begin{array}{l} \textbf{fresh}\ x_1 .. x_n \\ b = (b_1, .., b_n) \\ pat''_1 .. pat''_n = \textbf{duplicate}\ \_ b_1 .. b_n \end{array}}{E \vdash \_\ pat'_1 .. pat'_m \Rightarrow exp, \Pi : b \rightsquigarrow pat''_1 .. pat''_n\ pat'_1 .. pat'_m \Rightarrow exp, \Pi; b_1/x_1 .. b_n/x_n}\ \text{PHT\_WILD}$$

$$\frac{\begin{array}{l} \textbf{fresh}\ x_1 .. x_n \\ b = (b_1, .., b_n) \\ pat''_1 .. pat''_n = \textbf{duplicate}\ id\ b_1 .. b_n \end{array}}{E \vdash id\ pat'_1 .. pat'_m \Rightarrow exp, \Pi : b \rightsquigarrow pat''_1 .. pat''_n\ pat'_1 .. pat'_m \Rightarrow exp, \Pi; b_1/x_1 .. b_n/x_n}\ \text{PHT\_VAR}$$

$\boxed{E \vdash funcl_1\ \textbf{and}\ ...\ \textbf{and}\ funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def}$

$$\frac{\begin{array}{l} id_1 ... id_n \rightsquigarrow f \\ E \vdash (pat_1 \Rightarrow exp_1), ..., (pat_n \Rightarrow exp_n) : b/x \rightsquigarrow \Theta; \Phi; B; \Gamma; \Delta \vdash s : \tau \end{array}}{E \vdash id_1\ pat_1 = exp_1\ \textbf{and}\ ...\ \textbf{and}\ id_n\ pat_n = exp_n \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{function}\ f(x) = s}\ \text{CFL\_FUNCL}$$

$\boxed{E \vdash def \rightsquigarrow \Theta; \Phi; \Delta \vdash def_1, .., def_n}$

$$\frac{\begin{array}{l} E; \epsilon \vdash (typ_1, ..., typ_n); \textbf{snd}\ x \rightsquigarrow b; \phi \\ E; \epsilon \vdash typ; z \rightsquigarrow b_2; \phi_2 \end{array}}{E \vdash \textbf{val}\ (typ_1, ..., typ_n) \rightarrow typ_2\ \textbf{effect}\ effect\ id \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{val}\ f : (x : \textbf{unit} * b[\phi]) \rightarrow \{z : b_2 | \phi_2\}}\ \text{CDEF\_FUNSPE}$$

$$\frac{\begin{array}{l} typquant \rightsquigarrow kinded\_id_1 .. kinded\_id_m, n\_constraint \\ \textbf{is\_kid\_map}\ M, b, \textbf{fst}\ x, kinded\_id_1 .. kinded\_id_m \\ E; M \vdash n\_constraint \rightsquigarrow \phi \\ E; M \vdash (typ_1, ..., typ_n); \textbf{snd}\ x \rightsquigarrow b_1; \phi_1 \\ E; M \vdash typ; z \rightsquigarrow b_2; \phi_2 \end{array}}{E \vdash \textbf{val}\ typquant\ (\ \overline{typ_i}^{\,i \in 1...n}\ ) \rightarrow typ\ \textbf{effect}\ effect\ id \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{val}\ f : (x : b * b_1[\phi \wedge \phi_1]) \rightarrow \{z : b_2 | \phi_2\}}\ \text{CDEF\_}$$

$$\frac{\begin{array}{l} id \sim tid \\ E \vdash id_1 \rightsquigarrow \dot{ctor}_1, tid \quad ... \quad E \vdash id_n \rightsquigarrow \dot{ctor}_n, tid \\ E \vdash typ_1 \rightsquigarrow \tau_1 \quad ... \quad E \vdash typ_n \rightsquigarrow \tau_n \end{array}}{E \vdash \textbf{typedef}\ id = \textbf{const union}\ \{typ_1\ id_1;\ ...\ ;typ_n\ id_n\ ;^?\} \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{union}\ tid = \{\dot{ctor}_1 : \tau_1,\ ...\ , \dot{ctor}_n : \tau_n\}}$$

$$\frac{\begin{array}{l} id \sim tid \\ E \vdash id_1 \rightsquigarrow \dot{ctor}_1, tid \quad ... \quad E \vdash id_n \rightsquigarrow \dot{ctor}_n, tid \\ typquant \rightsquigarrow kinded\_id_1\ ..\ kinded\_id_m, n\_constraint \\ \textbf{is\_kid\_map}\ M, b, \textbf{fst}\ x, kinded\_id_1\ ..\ kinded\_id_m \\ E; M \vdash n\_constraint \rightsquigarrow \phi \\ E; M \vdash typ_1; \textbf{snd}\ z \rightsquigarrow b_1; \phi_1 \quad ... \quad E; M \vdash typ_1; \textbf{snd}\ z \rightsquigarrow b_n; \phi_n \end{array}}{E \vdash \textbf{typedef}\ id = \textbf{const union}\ typquant\{typ_1\ id_1;\ ...\ ;typ_n\ id_n\ ;^?\} \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{union}\ tid = \forall\, \beta.\{\dot{ctor}_1 : \{z : b * \}$$

$$\frac{\begin{array}{l} E \vdash id_1 \rightsquigarrow \dot{ctor}_1, tid \quad ... \quad E \vdash id_n \rightsquigarrow \dot{ctor}_n, tid \\ id \sim tid \end{array}}{E \vdash \textbf{typedef}\ id = \textbf{enumerate}\ \{id_1;\ ...\ ;id_n\ ;^?\} \rightsquigarrow \Theta; \Phi; \Delta \vdash \textbf{union}\ tid = \{\dot{ctor}_1 : \{z : \textbf{unit} | \top\},\ ...\ , \dot{ctor}_n : \{z : \textbf{u}$$

$$\frac{E \vdash funcl_1\ \textbf{and}\ ...\ \textbf{and}\ funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def}{E \vdash \textbf{function}\ rec\_opt\ \ effect\_opt\ funcl_1\ \textbf{and}\ ...\ \textbf{and}\ funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def}\ \text{CDEF\_FUNDEF}$$

$$\frac{\begin{array}{l} E \vdash \textbf{val}\ typquant\ typ\ id \rightsquigarrow \Theta; \Phi; \Delta \vdash def_1 \\ E \vdash funcl_1\ \textbf{and}\ ...\ \textbf{and}\ funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def_2 \end{array}}{E \vdash \textbf{function}\ rec\_opt\ typquant\ typ\ effect\_opt\ funcl_1\ \textbf{and}\ ...\ \textbf{and}\ funcl_n \rightsquigarrow \Theta; \Phi; \Delta \vdash def_1, def_2}\ \text{CDEF\_FUNDEF\_SP}$$

$$\frac{\begin{array}{l} E \vdash typ \rightsquigarrow \tau \\ id \sim u \end{array}}{E \vdash \textbf{register}\ effect\ effect'\ typ\ id \rightsquigarrow \Theta; \Phi; \Delta, u : \tau \vdash}\ \text{CDEF\_REGISTER}$$

$$\boxed{E \vdash def_1\ ..\ def_n \rightsquigarrow \Theta; \Phi \vdash def_1\ ..\ def_m}$$

$$\frac{E \vdash def \rightsquigarrow \Theta; \Phi \vdash def}{E \vdash def\ def_1\ ..\ def_n \rightsquigarrow \Theta; \Phi \vdash def\ def_1\ ..\ def_n}\ \text{CDEFS\_CONS}$$

```
Definition rules:         234 good    0 bad
Definition rule clauses:  756 good    0 bad
```