

Sail Manual

Kathryn E Gray, Gabriel Kerneis, Peter Sewell

October 25, 2016

Contents

1	Introduction	2
2	Tips for Writing Sail specifications	2
3	Sail syntax	3
4	Sail primitive types and functions	15
5	Sail type system	19
5.1	Internal type syntax	19
5.2	Type relations	25
6	Sail operational semantics {TODO}	53

1 Introduction

This is a manual describing the Sail specification language, its common library, compiler, interpreter and type system. However it is currently in early stages of being written, so questions to the developers are highly encouraged.

2 Tips for Writing Sail specifications

This section attempts to offer advice for writing Sail specifications that will work well with the Sail executable interpreter and compilers.

These tips use idiomatic Sail code; the Sail syntax is formally defined in following section.

Some tips might also be advice for good ways to specify instructions; this will come from a combination of users and Sail developers.

- Declare memory access functions as one read, one write announce, and one write value for each kind of access.

For basic user-mode instructions, there should be the need for only one memory read and two memory write function. These should each be declared using `val extern` and should have effect `wmem` and `rmem` accordingly.

Commonly, a memory read function will take as parameters a size (an number below 32) and an address and return a bit vector with length (8 * size). The sequential and concurrent interpreters both only read and write memory as a factor of bytes.

- Declare a default vector order

Vectors can be either decreasing or increasing, i.e. if we have a vector *a* with elements [1,2,3] then in an increasing specification the 1 is accessed with `a[0]` but with `a[2]` in a decreasing system. Early in your specification, it is beneficial to clarity to say default `Ord inc` or default `Ord dec`.

- Vectors don't necessarily begin indexing at 0 or n-1

Without any additional specification, a vector will begin indexing at 0 in an increasing spec and n-1 in a decreasing specification. A type declaration can reset this first position to any number.

Importantly, taking a slice of a vector does not reset the indexes. So if `a = [1,2,3,4]` in an increasing system the slice `a[2 .. 3]` generates the vector `[3,4]` and the 3 is indexed at 2 in either vector.

- Be precise in numeric types.

While Sail includes very wide types like `int` and `nat`, consider that for bounds checking, numeric operations, and and clear understanding, these really are unbounded quantities. If you know that a number in the specification will range only between 0 and 32, 0 and 4, -32 to 32 , it is better to use a specific range type such as `[|32|]`.

Similarly, if you don't know the range precisely, it may also be best to remain polymorphic and let Sail's type resolution work out bounds in a particular use rather than removing all bounds; to do this, use `[:'n:]` to say that it will polymorphically take some number.

- Use bit vectors for registers.

Sail the language will readily allow a register to store a value of any type. However, the Sail executable interpreter expects that it is simulating a uni-processor machine where all registers are bit vectors.

A vector of length one, such as a can read the element of a either with \mathbf{a} or $\mathbf{a}[0]$.

- Have functions named `decode` and `execute` to evaluate instructions.

The sail interpreter is hard-wired to look for functions with these names.

- Type annotations are necessary to read the contents of a register into a local variable.

The code $\mathbf{x} := \mathbf{GPR}[4]$, where `GPR` is a vector of general purpose registers, will store a local reference to the fourth general purpose register, not the contents of that register, i.e. this will not read the register. To read the register contents into a local variable, the type is required explicitly so $(\mathbf{bit}[64]) \mathbf{x} := \mathbf{GPR}[4]$ reads the register contents into \mathbf{x} . The type annotation may be on either side of the assignment.

3 Sail syntax

l	::=	Source location
$annot$::=	
id	::=	Identifier
		x
		$(\mathbf{deinfix} \ x)$ remove infix status
		bool M Built in type identifiers
		bit M
		unit M
		nat M
		string M
		range M
		atom M

		vector	M	
		list	M	
		set	M	
		reg	M	
		<i>to_num</i>	M	Built in function identifiers
		<i>to_vec</i>	M	
		msb	M	
<i>kid</i>	::=			variables with kind, ticked to differentiate from program variables
		' <i>x</i>		
<i>base_kind</i>	::=			base kind
		Type		kind of types
		Nat		kind of natural number size expressions
		Order		kind of vector order specifications
		Effect		kind of effect sets
<i>kind</i>	::=			kinds
		$base_kind_1 \rightarrow \dots \rightarrow base_kind_n$		
<i>nexp</i>	::=			expression of kind Nat, for vector sizes and origins
		<i>id</i>		identifier, bound by def Nat x = nexp
		<i>kid</i>		variable
		<i>num</i>		constant
		$nexp_1 * nexp_2$		product
		$nexp_1 + nexp_2$		sum
		$nexp_1 - nexp_2$		subtraction
		$2 * nexp$		exponential
		neg <i>nexp</i>		For internal use
		(<i>nexp</i>)	S	

<i>order</i>	$::=$ <i>kid</i> inc dec (<i>order</i>)		vector order specifications, of kind Order variable increasing (little-endian) decreasing (big-endian)
<i>base_effect</i>	$::=$ rreg wreg rmem wmem wmea wmv barr depend undef unspec nondet escape lset lret		effect read register write register read memory write memory signal effective address for writing memory write memory, sending only value memory barrier dynamic footprint undefined-instruction exception unspecified values nondeterminism from intra-instruction parallelism Tracking of expressions and functions that might call exit Local mutation happend; not user-writable Local return happened; not user-writable
<i>effect</i>	$::=$ <i>kid</i> $\{base_effect_1, \dots, base_effect_n\}$ pure $effect_1 \uplus \dots \uplus effect_n$	 M M	effect set, of kind Effects effect set sugar for empty effect set meta operation for combining sets of effects
<i>typ</i>	$::=$ -		Type expressions, of kind Type Unspecified type

		<i>id</i>	Defined type
		<i>kid</i>	Type variable
		$typ_1 \rightarrow typ_2$ effect <i>effect</i>	Function type (first-order only in user code)
		(typ_1, \dots, typ_n)	Tuple type
		$id\langle typ_arg_1, \dots, typ_arg_n \rangle$	type constructor application
		(typ)	S
		$[nexp]$	S sugar for range <0, <i>nexp</i> >
		$[nexp : nexp']$	S sugar for range < <i>nexp</i> , <i>nexp'</i> >
		$[: nexp :]$	S sugar for atom < <i>nexp</i> > which is special case of range < <i>nexp</i> , <i>nexp</i> >
		$typ[nexp]$	S sugar for vector indexed by [<i>nexp</i>]
		$typ[nexp : nexp']$	S sugar for vector indexed by [<i>nexp</i> .. <i>nexp'</i>]
		$typ[nexp < : nexp']$	S sugar for increasing vector indexed as above
		$typ[nexp > : nexp']$	S sugar for decreasing vector indexed as above
<i>typ_arg</i>	::=		Type constructor arguments of all kinds
		<i>nexp</i>	
		<i>typ</i>	
		<i>order</i>	
		<i>effect</i>	
<i>n_constraint</i>	::=		constraint over kind Nat
		$nexp = nexp'$	
		$nexp \geq nexp'$	
		$nexp \leq nexp'$	
		$kid \text{ IN } \{num_1, \dots, num_n\}$	
<i>kinded_id</i>	::=		optionally kind-annotated identifier
		<i>kid</i>	identifier
		<i>kind kid</i>	kind-annotated variable

<i>quant_item</i>	::=	Either a kinded identifier or a nexp constraint for a typquant
		<i>kinded_id</i> An optionally kinded identifier
		<i>n_constraint</i> A constraint for this type
<i>typquant</i>	::=	type quantifiers and constraints
		forall <i>quant_item</i> ₁ , ..., <i>quant_item</i> _n .
		sugar, omitting quantifier and constraints
<i>typschm</i>	::=	type scheme
		<i>typquant typ</i>
<i>name_scm_opt</i>	::=	Optional variable-naming-scheme specification for variables of defined type
		[name = <i>regex</i>]
<i>type_def</i>	::=	Type definition body
		typedef <i>id name_scm_opt</i> = <i>typschm</i> type abbreviation
		typedef <i>id name_scm_opt</i> = const struct <i>typquant</i> { <i>typ</i> ₁ <i>id</i> ₁ ; ...; <i>typ</i> _n <i>id</i> _n ; [?] }
		struct type definition
		typedef <i>id name_scm_opt</i> = const union <i>typquant</i> { <i>type_union</i> ₁ ; ...; <i>type_union</i> _n ; [?] }
		union type definition
		typedef <i>id name_scm_opt</i> = enumerate { <i>id</i> ₁ ; ...; <i>id</i> _n ; [?] }
		enumeration type definition
		typedef <i>id</i> = register bits [<i>nexp</i> : <i>nexp'</i>]{ <i>index_range</i> ₁ : <i>id</i> ₁ ; ...; <i>index_range</i> _n : <i>id</i> _n }
		register mutable bitfield type definition
<i>type_union</i>	::=	Type union constructors
		<i>id</i>

		<i>typ id</i>	
<i>index_range</i>	::=		index specification, for bitfields in register types
		<i>num</i>	single index
		<i>num</i> ₁ .. <i>num</i> ₂	index range
		<i>index_range</i> ₁ , <i>index_range</i> ₂	concatenation of index ranges
<i>lit</i>	::=		Literal constant
		()	() : unit
		bitzero	bitzero : bit
		bitone	bitone : bit
		true	true : bool
		false	false : bool
		<i>num</i>	natural number constant
		<i>hex</i>	bit vector constant, C-style
		<i>bin</i>	bit vector constant, C-style
		undefined	constant representing undefined values
		<i>string</i>	string constant
<i>;</i> [?]	::=		Optional semi-colon
		;	
<i>pat</i>	::=		Pattern
		<i>lit</i>	literal constant pattern
		-	wildcard
		(<i>pat</i> as <i>id</i>)	named pattern
		(<i>typ</i>) <i>pat</i>	typed pattern
		<i>id</i>	identifier
		<i>id</i> (<i>pat</i> ₁ , .., <i>pat</i> _{<i>n</i>})	union constructor pattern

		$\{fpat_1; \dots; fpat_n; ?\}$		struct pattern
		$[pat_1, \dots, pat_n]$		vector pattern
		$[num_1 = pat_1, \dots, num_n = pat_n]$		vector pattern (with explicit indices)
		$pat_1 : \dots : pat_n$		concatenated vector pattern
		(pat_1, \dots, pat_n)		tuple pattern
		$[pat_1, \dots, pat_n]$		list pattern
		(pat)	S	
$fpat$	$::=$			Field pattern
		$id = pat$		
exp	$::=$			Expression
		$\{exp_1; \dots; exp_n\}$		block
		nondet $\{exp_1; \dots; exp_n\}$		nondeterministic block, expressions evaluate in an unspecified order, or concurrently
		id		identifier
		lit		literal constant
		$(typ)exp$		cast
		$id(exp_1, \dots, exp_n)$		function application
		$id\ exp$	S	No extra parens needed when exp is a tuple
		$exp_1\ id\ exp_2$		infix function application
		(exp_1, \dots, exp_n)		tuple
		if exp_1 then exp_2 else exp_3		conditional
		if exp_1 then exp_2	S	
		foreach $(id\ \text{from}\ exp_1\ \text{to}\ exp_2\ \text{by}\ exp_3\ \text{in}\ order)\ exp_4$		loop
		foreach $(id\ \text{from}\ exp_1\ \text{to}\ exp_2\ \text{by}\ exp_3)\ exp_4$	S	
		foreach $(id\ \text{from}\ exp_1\ \text{to}\ exp_2)\ exp_3$	S	
		foreach $(id\ \text{from}\ exp_1\ \text{downto}\ exp_2\ \text{by}\ exp_3)\ exp_4$	S	
		foreach $(id\ \text{from}\ exp_1\ \text{downto}\ exp_2)\ exp_3$	S	
		$[exp_1, \dots, exp_n]$		vector (indexed from 0)
		$[num_1 = exp_1, \dots, num_n = exp_n\ opt_default]$		vector (indexed consecutively)

	<i>exp</i> [<i>exp'</i>]		vector access
	<i>exp</i> [<i>exp</i> ₁ .. <i>exp</i> ₂]		subvector extraction
	[<i>exp</i> with <i>exp</i> ₁ = <i>exp</i> ₂]		vector functional update
	[<i>exp</i> with <i>exp</i> ₁ : <i>exp</i> ₂ = <i>exp</i> ₃]		vector subrange update (with vector)
	<i>exp</i> : <i>exp</i> ₂		vector concatenation
	[<i>exp</i> ₁ , .., <i>exp</i> _{<i>n</i>}]		list
	<i>exp</i> ₁ :: <i>exp</i> ₂		cons
	{ <i>fexp</i> s}		struct
	{ <i>exp</i> with <i>fexp</i> s}		functional update of struct
	<i>exp.id</i>		field projection from struct
	switch <i>exp</i> { case <i>pexp</i> ₁ ... case <i>pexp</i> _{<i>n</i>} }		pattern matching
	let <i>bind</i> in <i>exp</i>		let expression
	<i>lexp</i> := <i>exp</i>		imperative assignment
	sizeof <i>nexp</i>		Expression to return the value of the <i>nexp</i> variable or expression at run time
	exit <i>exp</i>		expression to halt all current execution, potentially calling a system, trap, or interrupt handler with <i>exp</i>
	return <i>exp</i>		expression to end current function execution and return the value of <i>exp</i> from the function; this can be used
	assert (<i>exp</i> , <i>exp'</i>)		expression to halt with error, when the first expression is false, reporting the optional string as an error
	(<i>exp</i>)	S	
	(<i>annot</i>) <i>exp</i>		This is an internal cast, generated during type checking that will resolve into a syntactic cast after
	<i>annot</i>		This is an internal use for passing <i>nexp</i> information to library functions, postponed for constraint solving
	sizeof <i>annot</i>		For <i>sizeof</i> during type checking, to replace <i>nexp</i> with internal <i>n</i>
	<i>annot</i> , <i>annot'</i>		This is like the above but the user has specified an implicit parameter for the current function
	comment <i>string</i>		For generated unstructured comments
	comment <i>exp</i>		For generated structured comments
	let <i>lexp</i> = <i>exp</i> in <i>exp'</i>		This is an internal node for compilation that demonstrates the scope of a local mutable variable
	let <i>pat</i> = <i>exp</i> in <i>exp'</i>		This is an internal node, used to distinguished some introduced lets during processing from original ones
	<i>return_int</i> (<i>exp</i>)		For internal use to embed into monad definition
<i>lexp</i>	::=		lvalue expression
	<i>id</i>		identifier

	$ \quad id(exp_1, \dots, exp_n)$ $ \quad id \ exp$ $ \quad (typ)id$ $ \quad (lexp_0, \dots, lexp_n)$ $ \quad lexp[exp]$ $ \quad lexp[exp_1..exp_2]$ $ \quad lexp.id$	S set multiple at a time, a check will ensure it's not memory vector element subvector struct field
$fexp$	$::=$ $ \quad id = exp$	Field-expression
$fexps$	$::=$ $ \quad fexp_1; \dots; fexp_n; ?$	Field-expression list
$opt_default$	$::=$ $ $ $ \quad ; \textbf{default} = exp$	Optional default value for indexed vectors, to define a default value for any unspecified positions in a sparse map
$pexp$	$::=$ $ \quad pat \rightarrow exp$	Pattern match
$tannot_opt$	$::=$ $ \quad typquant \ typ$	Optional type annotation for functions
rec_opt	$::=$ $ $ $ \quad \textbf{rec}$	Optional recursive annotation for functions non-recursive recursive
$effect_opt$	$::=$ $ $	Optional effect annotation for functions sugar for empty effect set

		effect <i>effect</i>	
<i>funcl</i>	::=		Function clause
		<i>id pat = exp</i>	
<i>fundef</i>	::=		Function definition
		function <i>rec_opt tannot_opt effect_opt funcl₁ and ... and funcl_n</i>	
<i>letbind</i>	::=		Let binding
		let <i>typschm pat = exp</i>	value binding, explicit type (<i>pat</i> must be total)
		let <i>pat = exp</i>	value binding, implicit type (<i>pat</i> must be total)
<i>val_spec</i>	::=		Value type specification
		val <i>typschm id</i>	
		val extern <i>typschm id</i>	
		val extern <i>typschm id = string</i>	Specify the type and id of a function from Lem, where the string must provide an explicit path to the required function
<i>default_spec</i>	::=		Default kinding or typing assumption
		default <i>base_kind kid</i>	
		default Order <i>order</i>	
		default <i>typschm id</i>	
<i>scattered_def</i>	::=		Function and type union definitions that can be spread across a file. Each one must end in id
		scattered function <i>rec_opt tannot_opt effect_opt id</i>	scattered function definition header
		function clause <i>funcl</i>	scattered function definition clause
		scattered typedef <i>id name_scm_opt = const union typquant</i>	scattered union definition header
		union <i>id member type_union</i>	scattered union definition member

	end <i>id</i>	scattered definition end
<i>reg_id</i>	::= <i>id</i>	
<i>alias_spec</i>	::= <i>reg_id.id</i> <i>reg_id[exp]</i> <i>reg_id[exp..exp']</i> <i>reg_id : reg_id'</i>	Register alias expression forms. Other than where noted, each id must refer to an unaliased register of type <i>ve</i>
<i>dec_spec</i>	::= register <i>typ id</i> register alias <i>id = alias_spec</i> register alias <i>typ id = alias_spec</i>	Register declarations
<i>def</i>	::= <i>kind_def</i> <i>type_def</i> <i>fundef</i> <i>letbind</i> <i>val_spec</i> <i>default_spec</i> <i>scattered_def</i> <i>dec_spec</i> <i>dec_comm</i>	Top-level definition definition of named kind identifiers type definition function definition value definition top-level type constraint default kind and type assumptions scattered function and type definition register declaration generated comments
<i>defs</i>	::= <i>def₁ .. def_n</i>	Definition sequence

4 Sail primitive types and functions

<i>built_in_types</i>	<pre> ::= bit : Typ unit : Typ forall Nat '<i>n</i>. Nat '<i>m</i>. range <' <i>n</i>, '<i>m</i> >: Nat → Nat → Typ forall Nat '<i>n</i>. atom <' <i>n</i> >: Nat → Typ forall Nat '<i>n</i>, Nat '<i>m</i>, Order '<i>o</i>, Typ '<i>t</i>. vector <' <i>n</i>, '<i>m</i>, '<i>o</i>, '<i>t</i> >: Nat → Nat → Order → Typ forall Typ '<i>a</i>. option <' <i>a</i> >: Typ → Typ forall Typ '<i>t</i>. register <' <i>t</i> >: Typ → Typ forall Typ '<i>t</i>. reg <' <i>t</i> >: Typ → Typ forall Nat '<i>n</i>. implicit <' <i>n</i> >: Nat → Typ </pre>	<p>Type Kind</p> <p>singleton number, instead of range<' <i>n</i>, '<i>m</i> ></p> <p>internal reference cell To add to a function val specification ind</p>
<i>built_in_type_abbreviations</i>	<pre> ::= bool ⇒ bit nat ⇒ [0..<i>pos_infinity</i>] int ⇒ [<i>neg_infinity</i>..<i>pos_infinity</i>] uint8 ⇒ [0..<i>2 * 8</i>] uint16 ⇒ [0..<i>2 * 16</i>] uint32 ⇒ [0..<i>2 * 32</i>] uint64 ⇒ [0..<i>2 * 64</i>] </pre>	
<i>functions</i>	<pre> ::= val forall Typ '<i>a</i>. '<i>a</i> → unit : ignore val forall Typ '<i>a</i>. '<i>a</i> → option <' <i>a</i> > : Some val forall Typ '<i>a</i>. unit → option <' <i>a</i> > : None val ([: '<i>n</i> :], [: '<i>m</i> :]) → ['<i>n</i> + '<i>m</i>] : + val forall Nat '<i>n</i>. (bit ['<i>n</i>], bit ['<i>n</i>]) → bit ['<i>n</i>] : + val forall Nat '<i>n</i>. (bit ['<i>n</i>], bit ['<i>n</i>]) → (bit ['<i>n</i>], bit , bit) : + val forall Nat '<i>n</i>. (bit ['<i>n</i>], bit ['<i>n</i>]) → bit ['<i>n</i>] : +_s val forall Nat '<i>n</i>. (bit ['<i>n</i>], bit ['<i>n</i>]) → (bit ['<i>n</i>], bit , bit) : +_s </pre>	<p>Built-in functions: all have effect pure, all order polymorphic</p> <p>arithmetic addition unsigned vector addition unsigned vector addition with overflow, carry out signed vector addition signed vector addition with overflow, carry out</p>

```

| val (['n..'m], ['o..'p]) → ['n - 'o..'m - 'p] : -
| val forall Nat 'n.(bit ['n], bit ['n]) → bit ['n] : -
| val forall Nat 'n.(bit ['n], bit ['n]) → (bit ['n], bit , bit ) : -
| val forall Nat 'n.(bit ['n], bit ['n]) → bit ['n] : -s
| val forall Nat 'n.(bit ['n], bit ['n]) → (bit ['n], bit , bit ) : -s
| val (['n..'m], ['o..'p]) → ['n * 'o..'m * 'p] : *
| val forall Nat 'n.(bit ['n], bit ['n]) → bit [2 * 'n] : *
| val forall Nat 'n.(bit ['n], bit ['n]) → bit [2 * 'n] : *s
| val (['n..'m], ['1..'p]) → ['0..'p - 1] : mod
| val forall Nat 'n.(bit ['n], bit ['n]) → bit ['n] : mod
| val (['n..'m], ['1..'p]) → ['q..'r] : quot
| val forall Nat 'n, Nat 'm.(bit ['n], bit ['m]) → bit ['n] : quot
| val forall Nat 'n, Nat 'm.(bit ['n], bit ['m]) → bit ['n] : quot_s
| val forall Typ 'a, Nat 'n.(a ['n] → ['n] ) : length
| val forall Typ 'a, Nat 'n, Nat 'm, n ≤ m.(implicit ('m), a ['n]) → a ['m] : mask
| val forall Nat 'n.(bit ['n], bit ['n]) → bit :≡
| val forall Typ 'a, Typ 'b.(a, b) → bit :≡
| val forall Typ 'a, Typ 'b.(a, b) → bit :!=
| val (['n..'m], ['o..'p]) → bit : <
| val forall Nat 'n.(bit ['n], bit ['n]) → bit : <
| val forall Nat 'n.(bit ['n], bit ['n]) → bit :< s
| val (['n..'m], ['o..'p]) → bit : >
| val forall Nat 'n.(bit ['n], bit ['n]) → bit : >
| val forall Nat 'n.(bit ['n], bit ['n]) → bit :> s
| val (['n..'m], ['o..'p]) → bit : ≤
| val forall Nat 'n.(bit ['n], bit ['n]) → bit : ≤
| val forall Nat 'n.(bit ['n], bit ['n]) → bit :<= s
| val (['n..'m], ['o..'p]) → bit : ≥
| val forall Nat 'n.(bit ['n], bit ['n]) → bit : ≥

```

arithmetic subtraction

unsigned vector subtraction

unsigned vector subtraction with overflow, carry out

signed vector subtraction

signed vector subtraction with overflow, carry out

arithmetic multiplication

unsigned vector multiplication

signed vector multiplication

arithmetic modulo

unsigned vector modulo

arithmetic integer division

unsigned vector division

signed vector division

reduce size of vector, dropping MSBits. Type system supplies implicit 1

vector equality

unsigned less than

unsigned greater than

unsigned less than or eq

unsigned greater than or eq

	val forall Nat <i>'n</i> .(bit [<i>'n</i>], bit [<i>'n</i>]) → bit :>= <i>_s</i>	
	val bit → bit :	bit negation
	val forall Nat <i>'n</i> . bit [<i>'n</i>] → bit [<i>'n</i>] :	bitwise negation
	val (bit , bit) → bit :	bitwise or
	val forall Nat <i>'n</i> .(bit [<i>'n</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] :	
	val (bit , bit) → bit :&	bitwise and
	val forall Nat <i>'n</i> .(bit [<i>'n</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] :&	
	val (bit , bit) → bit :↑	bitwise xor
	val forall Nat <i>'n</i> .(bit [<i>'n</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] :↑	
	val forall Nat <i>'n</i> .(bit , [<i>'n</i>]) → bit [<i>'n</i>] :↑↑	duplicate bit into a vector
	val forall Nat <i>'n</i> , Nat <i>'m</i> , <i>'m</i> ≤' <i>n</i> .(bit [<i>'n</i>], [<i>'m</i>]) → bit [<i>'n</i>] :<<	left shift
	val forall Nat <i>'n</i> , Nat <i>'m</i> , <i>'m</i> ≤' <i>n</i> .(bit [<i>'n</i>], [<i>'m</i>]) → bit [<i>'n</i>] :>>	right shift
	val forall Nat <i>'n</i> , Nat <i>'m</i> , <i>'m</i> ≤' <i>n</i> .(bit [<i>'n</i>], [<i>'m</i>]) → bit [<i>'n</i>] :<<<	rotate

functions_with_coercions

::=

	val forall Nat <i>'n</i> .(<i>bit</i> [<i>'n</i>], bit [<i>'n</i>]) → [<i>2**'n</i>] : +
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → bit [<i>'n</i>] : +
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .([<i>'o..'p</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] : +
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → [<i>'o..'p</i> + 2 * <i>'n</i>] : +
	val forall Nat <i>'n</i> (bit [<i>'n</i>], bit) → bit [<i>'n</i>] : +
	val forall Nat <i>'n</i> (bit , bit [<i>'n</i>]) → bit [<i>'n</i>] : +
	val forall Nat <i>'n</i> .(<i>bit</i> [<i>'n</i>], bit [<i>'n</i>]) → [<i>2**'n</i>] : +_s
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → bit [<i>'n</i>] : +_s
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .([<i>'o..'p</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] : +_s
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → [<i>'o..'p</i> + 2 * <i>'n</i>] : +_s
	val forall Nat <i>'n</i> (bit [<i>'n</i>], bit) → bit [<i>'n</i>] : +_s
	val forall Nat <i>'n</i> (bit , bit [<i>'n</i>]) → bit [<i>'n</i>] : +_s
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → bit [<i>'n</i>] : -
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .([<i>'o..'p</i>], bit [<i>'n</i>]) → bit [<i>'n</i>] : -
	val forall Nat <i>'n</i> , Nat <i>'o</i> , Nat <i>'p</i> .(bit [<i>'n</i>], [<i>'o..'p</i>]) → [<i>'o..'p</i> + 2 * <i>'n</i>] : -

5 Sail type system

5.1 Internal type syntax

k	$::=$	Internal kinds
	K_Typ	
	K_Nat	
	K_Ord	
	K_Efct	
	$K_Lam(k_0 \dots k_n \rightarrow k')$	
	K_infer	Representing an unknown kind, inferred by context
t, u	$::=$	Internal types
	x	
	$'x$	
	$t_1 \rightarrow t_2 \text{ effect}$	
	(t_1, \dots, t_n)	
	$x\langle t_args \rangle$	
	$t \mapsto t_1$	
	register $\langle t_arg \rangle$	S
	range $\langle ne \ ne' \rangle$	S
	atom $\langle ne \rangle$	S
	vector $\langle ne \ ne' \ order \ t \rangle$	S
	list $\langle t \rangle$	S
	reg $\langle t \rangle$	S
	implicit $\langle ne \rangle$	S
	bit	S
	<i>string</i>	S
	unit	S
	$t[t_arg_1/tid_1 \dots t_arg_n/tid_n]$	M

<i>optx</i>	::=	
		<i>x</i>
<i>tag</i>	::=	Data indicating where the identifier arises and thus information necessary in compilation
		None
		Intro Denotes an assignment and lexp that introduces a binding
		Set Denotes an expression that mutates a local variable
		Tuple Denotes an assignment with a tuple lexp
		Global Globally let-bound or enumeration based value/variable
		Ctor Data constructor from a type union
		Extern <i>optx</i> External function, specied only with a val statement
		Default Type has come from default declaration, identifier may not be bound locally
		Spec
		Enum <i>num</i>
		Alias
		<i>Unknown_pathoptx</i> Tag to distinguish an unknown path from a non-analysis non deterministic path
<i>ne</i>	::=	internal numeric expressions
		<i>x</i>
		' <i>x</i>
		<i>num</i>
		infinity
		<i>ne</i> ₁ * <i>ne</i> ₂
		<i>ne</i> ₁ + ... + <i>ne</i> _{<i>n</i>}
		<i>ne</i> ₁ − <i>ne</i> ₂
		2 ** <i>ne</i>
		(− <i>ne</i>)
		zero S
		one S

		bitlength (<i>bin</i>)	M	
		bitlength (<i>hex</i>)	M	
		count (<i>num</i> ₀ ... <i>num</i> _{<i>i</i>})	M	
		length (<i>pat</i> ₁ ... <i>pat</i> _{<i>n</i>})	M	
		length (<i>exp</i> ₁ ... <i>exp</i> _{<i>n</i>})	M	
<i>t_arg</i>	::=			Argument to type constructors
		<i>t</i>		
		<i>ne</i>		
		<i>effect</i>		
		<i>order</i>		
		fresh	M	
<i>t_args</i>	::=			Arguments to type constructors
		<i>t_arg</i> ₁ ... <i>t_arg</i> _{<i>n</i>}		
<i>nec</i>	::=			Numeric expression constraints
		<i>ne</i> ≤ <i>ne'</i>		
		<i>ne</i> = <i>ne'</i>		
		<i>ne</i> ≥ <i>ne'</i>		
		' <i>x</i> IN { <i>num</i> ₁ , ..., <i>num</i> _{<i>n</i>} }		
		<i>nec</i> ₀ .. <i>nec</i> _{<i>n</i>} → <i>nec'</i> ₀ ... <i>nec'</i> _{<i>m</i>}		
		<i>nec</i> ₀ ... <i>nec</i> _{<i>n</i>}		
Σ^N	::=			nexp constraint lists
		{ <i>nec</i> ₁ , ..., <i>nec</i> _{<i>n</i>} }		
		$\Sigma^N_1 \uplus \dots \uplus \Sigma^N_n$	M	
		consistent_increase <i>ne</i> ₁ <i>ne'</i> ₁ ... <i>ne</i> _{<i>n</i>} <i>ne'</i> _{<i>n</i>}	M	Generates constraints from pairs of constraints, where the first of each pair is always larger than the second
		consistent_decrease <i>ne</i> ₁ <i>ne'</i> ₁ ... <i>ne</i> _{<i>n</i>} <i>ne'</i> _{<i>n</i>}	M	Generates constraints from pairs of constraints, where the first of each pair is always smaller than the second
		resolve (Σ^N)		

E^D	$::=$ $\mid \langle E^K, E^A, E^R, E^E \rangle$ $\mid \epsilon$ $\mid E^D \uplus E^{D'}$	Environments storing top level information, such as defined abbreviations, records, enumerations, and kinds
$kinf$	$::=$ $\mid k$ $\mid k \text{ \textbf{default} }$	Whether a kind is default or from a local binding
tid	$::=$ $\mid id$ $\mid kid$	A type identifier or type variable
E^K	$::=$ $\mid \{tid_1 \mapsto kinf_1, \dots, tid_n \mapsto kinf_n\}$ $\mid E^K_1 \uplus \dots \uplus E^K_n$ $\mid E^K \setminus E^K_1 \dots E^K_n$	Kind environments M In a unioning kinf, k default u k results in k (i.e. the default is locally forgotten) M
$tinf$	$::=$ $\mid t$ $\mid E^K, \Sigma^N, tag, t$	Type variables, type, and constraints, bound to an identifier
E^A	$::=$ $\mid \{tid_1 \mapsto tinf_1, \dots, tid_n \mapsto tinf_n\}$ $\mid E^A_1 \uplus \dots \uplus E^A_n$	
$field_typs$	$::=$ $\mid id_1 : t_1, \dots, id_n : t_n$	Record fields
E^R	$::=$	Record environments

	$\begin{array}{l} \quad \{\{field_types_1\} \mapsto \mathit{tinf}_1, \dots, \{field_types_n\} \mapsto \mathit{tinf}_n\} \\ \quad E_1^R \uplus \dots \uplus E_n^R \end{array}$	M	
$enumerate_map$	$\begin{array}{l} ::= \\ \quad \{num_1 \mapsto id_1 \dots num_n \mapsto id_n\} \end{array}$		
E^E	$\begin{array}{l} ::= \\ \quad \{t_1 \mapsto enumerate_map_1, \dots, t_n \mapsto enumerate_map_n\} \\ \quad E_1^E \uplus \dots \uplus E_n^E \end{array}$		Enumeration environments
E^T	$\begin{array}{l} ::= \\ \quad \{id_1 \mapsto \mathit{tinf}_1, \dots, id_n \mapsto \mathit{tinf}_n\} \\ \quad \{id \mapsto \mathbf{overload} \mathit{tinf} \mathit{conformsto} : \mathit{tinf}_1, \dots, \mathit{tinf}_n\} \\ \quad (E_1^T \uplus \dots \uplus E_n^T) \\ \quad \uplus E_1^T \dots E_n^T \\ \quad E^T \setminus id_1 \dots id_n \\ \quad (E_1^T \cap \dots \cap E_n^T) \\ \quad \cap E_1^T \dots E_n^T \end{array}$	M M M M M	Type environments
ts	$\begin{array}{l} ::= \\ \quad t_1, \dots, t_n \end{array}$		
E	$\begin{array}{l} ::= \\ \quad \langle E^T, E^D \rangle \\ \quad \epsilon \\ \quad E \uplus E' \end{array}$		Definition environment and lexical environment
I	$\begin{array}{l} ::= \\ \quad \langle \Sigma^N, effect \rangle \\ \quad I_\epsilon \end{array}$		Information given by type checking an expression Empty constraints, effect

		$I_1 \uplus I_2$	
		$I_1 \uplus \dots \uplus I_n$	Unions the constraints and effect
<i>formula</i>	::=		
		<i>judgement</i>	
		$formula_1 \dots formula_n$	
		$E^K(tid) \triangleright kinf$	Kind lookup
		$E^A(tid) \triangleright tinf$	
		$E^A(tid) \triangleright ne$	
		$E^T(id) \triangleright tinf$	Type lookup
		$E^T(id) \triangleright \mathbf{overload} \ tinf : tinf_1 \dots tinf_n$	Overloaded type lookup
		$E^K(tid) < - k$	Update the kind associated with id to k
		$E^R(id_0 \dots id_n) \triangleright t, ts$	Record lookup
		$E^R(t) \triangleright id_0 : t_0 \dots id_n : t_n$	Record loopup by type
		$E^E(t) \triangleright enumerate_map$	Enumeration lookup by type
		$\mathbf{dom}(E^T_1) \cap \mathbf{dom}(E^T_2) = \emptyset$	
		$\mathbf{dom}(E^K_1) \cap \mathbf{dom}(E^K_2) = \emptyset$	
		$\mathbf{disjoint} \ \mathbf{doms}(E^T_1, \dots, E^T_n)$	Pairwise disjoint domains
		$id \notin \mathbf{dom}(E^K)$	
		$id \notin \mathbf{dom}(E^T)$	
		$id_0 : t_0 \dots id_n : t_n \subset id'_0 : t'_0 \dots id'_i : t'_i$	
		$num_1 < \dots < num_n$	
		$num_1 > \dots > num_n$	
		$exp_1 \equiv exp_2$	
		$E^K_1 \equiv E^K_2$	
		$E^K_1 \approx E^K_2$	
		$E^T_1 \equiv E^T_2$	
		$E^R_1 \equiv E^R_2$	
		$E^E_1 \equiv E^E_2$	
		$E^D_1 \equiv E^D_2$	

	$E_1 \equiv E_2$
	$\Sigma^N_1 \equiv \Sigma^N_2$
	$id \equiv' id$
	$x_1 \neq x_2$
	$lit_1 \neq lit_2$
	$I_1 \equiv I_2$
	$effect_1 \equiv effect_2$
	$t_1 \equiv t_2$
	$ne \equiv ne'$
	$kid \equiv fresh_kid(E^D)$

5.2 Type relations

$\boxed{E^K \vdash_t t \mathbf{ok}}$ Well-formed types

$$\begin{array}{c}
\frac{E^K('x) \triangleright K_Typ}{E^K \vdash_t 'x \mathbf{ok}} \quad \text{CHECK_T_VAR} \\
\\
\frac{E^K('x) \triangleright K_infer \quad E^K('x) < -|K_Typ}{E^K \vdash_t 'x \mathbf{ok}} \quad \text{CHECK_T_VARINFER} \\
\\
\frac{E^K \vdash_t t_1 \mathbf{ok} \quad E^K \vdash_t t_2 \mathbf{ok} \quad E^K \vdash_e effect \mathbf{ok}}{E^K \vdash_t t_1 \rightarrow t_2 effect \mathbf{ok}} \quad \text{CHECK_T_FN} \\
\\
\frac{E^K \vdash_t t_1 \mathbf{ok} \quad \dots \quad E^K \vdash_t t_n \mathbf{ok}}{E^K \vdash_t (t_1, \dots, t_n) \mathbf{ok}} \quad \text{CHECK_T_TUP} \\
\\
\frac{E^K(x) \triangleright K_Lam(k_1 \dots k_n \rightarrow K_Typ) \quad E^K, k_1 \vdash t_arg_1 \mathbf{ok} \quad \dots \quad E^K, k_n \vdash t_arg_n \mathbf{ok}}{E^K \vdash_t x \langle t_arg_1 \dots t_arg_n \rangle \mathbf{ok}} \quad \text{CHECK_T_APP}
\end{array}$$

$\boxed{E^K \vdash_e effect \mathbf{ok}}$ Well-formed effects

$$\begin{array}{c}
\frac{E^K('x) \triangleright K_Efct}{E^K \vdash_e 'x \mathbf{ok}} \quad \text{CHECK_EF_VAR} \\
\\
\frac{E^K('x) \triangleright K_infer \quad E^K('x) < -|K_Efct}{E^K \vdash_e 'x \mathbf{ok}} \quad \text{CHECK_EF_VARINFER} \\
\\
\frac{}{E^K \vdash_e \{base_effect_1, \dots, base_effect_n\} \mathbf{ok}} \quad \text{CHECK_EF_SET}
\end{array}$$

$E^K \vdash_n ne \mathbf{ok}$

Well-formed numeric expressions

$$\begin{array}{c}
\frac{E^K(x) \triangleright K_Nat}{E^K \vdash_n x \mathbf{ok}} \quad \text{CHECK_N_ID} \\
\\
\frac{E^K('x) \triangleright K_Nat}{E^K \vdash_n 'x \mathbf{ok}} \quad \text{CHECK_N_VAR} \\
\\
\frac{E^K('x) \triangleright K_infer \quad E^K('x) < -|K_Nat}{E^K \vdash_n 'x \mathbf{ok}} \quad \text{CHECK_N_VARINFER} \\
\\
\frac{}{E^K \vdash_n num \mathbf{ok}} \quad \text{CHECK_N_NUM} \\
\\
\frac{E^K \vdash_n ne_1 \mathbf{ok} \quad E^K \vdash_n ne_2 \mathbf{ok}}{E^K \vdash_n ne_1 + ne_2 \mathbf{ok}} \quad \text{CHECK_N_SUM} \\
\\
\frac{E^K \vdash_n ne_1 \mathbf{ok} \quad E^K \vdash_n ne_2 \mathbf{ok}}{E^K \vdash_n ne_1 - ne_2 \mathbf{ok}} \quad \text{CHECK_N_MINUS} \\
\\
\frac{E^K \vdash_n ne_1 \mathbf{ok} \quad E^K \vdash_n ne_2 \mathbf{ok}}{E^K \vdash_n ne_1 * ne_2 \mathbf{ok}} \quad \text{CHECK_N_MULT} \\
\\
\frac{E^K \vdash_n ne \mathbf{ok}}{E^K \vdash_n 2 ** ne \mathbf{ok}} \quad \text{CHECK_N_EXP}
\end{array}$$

$$E^K \vdash_o \text{order } \mathbf{ok}$$

Well-formed numeric expressions

$$\frac{E^K('x) \triangleright K_Ord}{E^K \vdash_o 'x \mathbf{ok}} \quad \text{CHECK_ORD_VAR}$$

$$\frac{E^K('x) \triangleright K_infer \quad E^K('x) < -|K_Ord}{E^K \vdash_o 'x \mathbf{ok}} \quad \text{CHECK_ORD_VARINFER}$$

$$\frac{}{E^K \vdash_o \mathbf{inc ok}} \quad \text{CHECK_ORD_INC}$$

$$\frac{}{E^K \vdash_o \mathbf{dec ok}} \quad \text{CHECK_ORD_DEC}$$

$$E^K, k \vdash t_arg \mathbf{ok}$$

Well-formed type arguments kind check matching the application type variable

$$\frac{E^K \vdash_t t \mathbf{ok}}{E^K, K_Typ \vdash t \mathbf{ok}} \quad \text{CHECK_TARGS_TYP}$$

$$\frac{E^K \vdash_e \text{effect } \mathbf{ok}}{E^K, K_Efct \vdash \text{effect } \mathbf{ok}} \quad \text{CHECK_TARGS_EFF}$$

$$\frac{E^K \vdash_n \text{ne } \mathbf{ok}}{E^K, K_Nat \vdash \text{ne } \mathbf{ok}} \quad \text{CHECK_TARGS_NAT}$$

$$\frac{E^K \vdash_o \text{order } \mathbf{ok}}{E^K, K_Ord \vdash \text{order } \mathbf{ok}} \quad \text{CHECK_TARGS_ORD}$$

$$E^K \vdash \text{kind } \rightsquigarrow k$$

$$\frac{}{E^K \vdash \mathbf{Type} \rightsquigarrow K_Typ} \quad \text{CONVERT_KIND_TYP}$$

$$E^D \vdash \text{quant_item } \rightsquigarrow E^K_1, \Sigma^N$$

Convert source quantifiers to kind environments and constraints

$$\frac{E^K \vdash \text{kind } \rightsquigarrow k}{\langle E^K, E^A, E^R, E^E \rangle \vdash \text{kind } 'x \rightsquigarrow \{ 'x \mapsto k \}, \{ \}} \quad \text{CONVERT_QUANTS_KIND}$$

$$\frac{E^K('x) \triangleright k}{\langle E^K, E^A, E^R, E^E \rangle \vdash 'x \rightsquigarrow \{'x \mapsto k\}, \{\}} \quad \text{CONVERT_QUANTS_NOKIND}$$

$$\frac{\begin{array}{c} \vdash \text{next}_1 \rightsquigarrow ne_1 \\ \vdash \text{next}_2 \rightsquigarrow ne_2 \end{array}}{E^D \vdash \text{next}_1 = \text{next}_2 \rightsquigarrow \{\}, \{ne_1 = ne_2\}} \quad \text{CONVERT_QUANTS_EQ}$$

$$\frac{\begin{array}{c} \vdash \text{next}_1 \rightsquigarrow ne_1 \\ \vdash \text{next}_2 \rightsquigarrow ne_2 \end{array}}{E^D \vdash \text{next}_1 \geq \text{next}_2 \rightsquigarrow \{\}, \{ne_1 \geq ne_2\}} \quad \text{CONVERT_QUANTS_GTEQ}$$

$$\frac{\begin{array}{c} \vdash \text{next}_1 \rightsquigarrow ne_1 \\ \vdash \text{next}_2 \rightsquigarrow ne_2 \end{array}}{E^D \vdash \text{next}_1 \leq \text{next}_2 \rightsquigarrow \{\}, \{ne_1 \leq ne_2\}} \quad \text{CONVERT_QUANTS_LTEQ}$$

$$\frac{}{E^D \vdash 'x \text{ IN } \{num_1, \dots, num_n\} \rightsquigarrow \{\}, \{'x \text{ IN } \{num_1, \dots, num_n\}\}} \quad \text{CONVERT_QUANTS_IN}$$

$\boxed{E^D \vdash \text{typeschm} \rightsquigarrow t, E^K, \Sigma^N}$ Convert source types with typeschemes to internal types and kind environments

$$\frac{E^D \vdash \text{typ} \rightsquigarrow t}{E^D \vdash \text{typ} \rightsquigarrow t, \{\}, \{\}} \quad \text{CONVERT_TYPSCHEM_NOQUANT}$$

$$\frac{\begin{array}{l} E^D \vdash \text{quant_item}_1 \rightsquigarrow E^K_1, \Sigma^N_1 \quad \dots \quad E^D \vdash \text{quant_item}_n \rightsquigarrow E^K_n, \Sigma^N_n \\ E^K \equiv E^K_1 \uplus \dots \uplus E^K_n \\ E^D \uplus \langle E^K, \{\}, \{\}, \{\} \rangle \vdash \text{typ} \rightsquigarrow t \end{array}}{E^D \vdash \mathbf{forall} \text{ quant_item}_1, \dots, \text{quant_item}_n. \text{typ} \rightsquigarrow t, E^K, \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n} \quad \text{CONVERT_TYPSCHEM_QUANT}$$

$\boxed{E^D \vdash \text{typ} \rightsquigarrow t}$ Convert source types to internal types

$$\frac{E^K('x) \triangleright K_Typ}{\langle E^K, E^A, E^R, E^E \rangle \vdash 'x \rightsquigarrow 'x} \quad \text{CONVERT_TYP_VAR}$$

$$\frac{E^K(x) \triangleright K_Typ}{\langle E^K, E^A, E^R, E^E \rangle \vdash x \rightsquigarrow x} \quad \text{CONVERT_TYP_ID}$$

$$\frac{\begin{array}{c} E^D \vdash \text{typ}_1 \rightsquigarrow t_1 \\ E^D \vdash \text{typ}_2 \rightsquigarrow t_2 \end{array}}{E^D \vdash \text{typ}_1 \rightarrow \text{typ}_2 \mathbf{effect} \rightsquigarrow t_1 \rightarrow t_2 \mathbf{effect}} \quad \text{CONVERT_TYP_FN}$$

$$\frac{E^D \vdash typ_1 \rightsquigarrow t_1 \quad \dots \quad E^D \vdash typ_n \rightsquigarrow t_n}{E^D \vdash (typ_1, \dots, typ_n) \rightsquigarrow (t_1, \dots, t_n)} \quad \text{CONVERT_TYP_TUP}$$

$$\frac{\begin{array}{l} E^K(x) \triangleright K_Lam(k_1 .. k_n \rightarrow K_Typ) \\ \langle E^K, E^A, E^R, E^E \rangle, k_1 \vdash typ_arg_1 \rightsquigarrow t_arg_1 \quad \dots \quad \langle E^K, E^A, E^R, E^E \rangle, k_n \vdash typ_arg_n \rightsquigarrow t_arg_n \end{array}}{\langle E^K, E^A, E^R, E^E \rangle \vdash x \langle typ_arg_1, \dots, typ_arg_n \rangle \rightsquigarrow x \langle t_arg_1, \dots, t_arg_n \rangle} \quad \text{CONVERT_TYP_APP}$$

$$\boxed{E^D, k \vdash typ_arg \rightsquigarrow t_arg} \quad \text{Convert source type arguments to internals}$$

$$\frac{E^D \vdash typ \rightsquigarrow t}{E^D, K_Typ \vdash typ \rightsquigarrow t} \quad \text{CONVERT_TARG_TYP}$$

$$\boxed{\vdash nexp \rightsquigarrow ne} \quad \text{Convert and normalize numeric expressions}$$

$$\frac{}{\vdash x \rightsquigarrow x} \quad \text{CONVERT_NEXP_ID}$$

$$\frac{}{\vdash 'x \rightsquigarrow 'x} \quad \text{CONVERT_NEXP_VAR}$$

$$\frac{}{\vdash num \rightsquigarrow num} \quad \text{CONVERT_NEXP_NUM}$$

$$\frac{\begin{array}{l} \vdash nexp_1 \rightsquigarrow ne_1 \\ \vdash nexp_2 \rightsquigarrow ne_2 \end{array}}{\vdash nexp_1 * nexp_2 \rightsquigarrow ne_1 * ne_2} \quad \text{CONVERT_NEXP_MULT}$$

$$\frac{\begin{array}{l} \vdash nexp_1 \rightsquigarrow ne_1 \\ \vdash nexp_2 \rightsquigarrow ne_2 \end{array}}{\vdash nexp_1 + nexp_2 \rightsquigarrow ne_1 + ne_2} \quad \text{CONVERT_NEXP_ADD}$$

$$\frac{\begin{array}{l} \vdash nexp_1 \rightsquigarrow ne_1 \\ \vdash nexp_2 \rightsquigarrow ne_2 \end{array}}{\vdash nexp_1 - nexp_2 \rightsquigarrow ne_1 - ne_2} \quad \text{CONVERT_NEXP_SUB}$$

$$\frac{\vdash nexp \rightsquigarrow ne}{\vdash 2 * nexp \rightsquigarrow 2 ** ne} \quad \text{CONVERT_NEXP_EXP}$$

$$\boxed{E^D \vdash_n ne \approx ne'}$$

$$\begin{array}{c}
\frac{E^K \vdash_n ne \mathbf{ok}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_n ne \approx ne} \quad \text{CONFORMS_TO_NE_REFL} \\
\\
\frac{\frac{E^D \vdash_n ne_1 \approx ne_2}{E^D \vdash_n ne_2 \approx ne_3}}{E^D \vdash_n ne_1 \approx ne_3} \quad \text{CONFORMS_TO_NE_TRANS} \\
\\
\frac{E^D \vdash_n ne_2 \approx ne_1}{E^D \vdash_n ne_1 \approx ne_2} \quad \text{CONFORMS_TO_NE_ASSOC} \\
\\
\frac{\frac{E^A(x) \triangleright ne}{\langle E^K, E^A, E^R, E^E \rangle \vdash_n ne \approx ne'}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_n x \approx ne'} \quad \text{CONFORMS_TO_NE_ABBREV} \\
\\
\frac{num \equiv num'}{E^D \vdash_n num \approx num'} \quad \text{CONFORMS_TO_NE_CONSTANTS} \\
\\
\frac{}{E^D \vdash_n ne \approx ne'} \quad \text{CONFORMS_TO_NE_REST}
\end{array}$$

$\boxed{E^D \vdash t \approx t'}$ Relate t and t' when t can be used where t' is expected without consideration for non-constant nats

$$\begin{array}{c}
\frac{E^K \vdash_t t \mathbf{ok}}{\langle E^K, E^A, E^R, E^E \rangle \vdash t \approx t} \quad \text{CONFORMS_TO_REFL} \\
\\
\frac{\frac{E^D \vdash t_1 \approx t_2}{E^D \vdash t_2 \approx t_3}}{E^D \vdash t_1 \approx t_3} \quad \text{CONFORMS_TO_TRANS} \\
\\
\frac{}{E^D \vdash 'x \approx t} \quad \text{CONFORMS_TO_VAR} \\
\\
\frac{}{E^D \vdash t \approx 'x} \quad \text{CONFORMS_TO_VAR2} \\
\\
\frac{\frac{E^A(x) \triangleright u}{\langle E^K, E^A, E^R, E^E \rangle \vdash u \approx t}}{\langle E^K, E^A, E^R, E^E \rangle \vdash x \approx t} \quad \text{CONFORMS_TO_ABBREV}
\end{array}$$

$$\begin{array}{c}
\frac{E^A(x) \triangleright u \quad \langle E^K, E^A, E^R, E^E \rangle \vdash t \approx u}{\langle E^K, E^A, E^R, E^E \rangle \vdash t \approx x} \text{ CONFORMS_TO_ABBREV2} \\
\\
\frac{E^D \vdash t_1 \approx u_1 \quad \dots \quad E^D \vdash t_n \approx u_n}{E^D \vdash (t_1, \dots, t_n) \approx (u_1, \dots, u_n)} \text{ CONFORMS_TO_TUP} \\
\\
\frac{E^K(x) \triangleright K_Lam(k_1 \dots k_n \rightarrow K_Typ) \quad \langle E^K, E^A, E^R, E^E \rangle, k_1 \vdash t_arg_1 \approx t_arg'_1 \quad \dots \quad \langle E^K, E^A, E^R, E^E \rangle, k_n \vdash t_arg_n \approx t_arg'_n}{\langle E^K, E^A, E^R, E^E \rangle \vdash x \langle t_arg_1 \dots t_arg_n \rangle \approx x \langle t_arg'_1 \dots t_arg'_n \rangle} \text{ CONFORMS_TO_APP} \\
\\
\frac{}{E^D \vdash \mathbf{atom} \langle ne \rangle \approx \mathbf{range} \langle ne_1 \ ne_2 \rangle} \text{ CONFORMS_TO_ATOM} \\
\\
\frac{}{E^D \vdash \mathbf{range} \langle ne_1 \ ne_2 \rangle \approx \mathbf{atom} \langle ne \rangle} \text{ CONFORMS_TO_ATOM2} \\
\\
\frac{x' \neq x \quad E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, tag, u \quad \langle E^K, E^A, E^R, E^E \rangle \vdash x \langle t_arg_1 \dots t_arg_n \rangle \approx u[t_arg'_1/tid_1 \dots t_arg'_m/tid_m]}{\langle E^K, E^A, E^R, E^E \rangle \vdash x \langle t_arg_1 \dots t_arg_n \rangle \approx x' \langle t_arg'_1 \dots t_arg'_m \rangle} \text{ CONFORMS_TO_APPABBREV} \\
\\
\frac{x' \neq x \quad E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u \quad \langle E^K, E^A, E^R, E^E \rangle \vdash u[t_arg_1/tid_1 \dots t_arg_n/tid_n] \approx x \langle t_arg'_1 \dots t_arg'_m \rangle}{\langle E^K, E^A, E^R, E^E \rangle \vdash x' \langle t_arg_1 \dots t_arg_n \rangle \approx x \langle t_arg'_1 \dots t_arg'_m \rangle} \text{ CONFORMS_TO_APPABBREV2} \\
\\
\frac{E^D \vdash t \approx u}{E^D \vdash \mathbf{register} \langle t \rangle \approx u} \text{ CONFORMS_TO_REGISTER} \\
\\
\frac{E^D \vdash t \approx u}{E^D \vdash \mathbf{reg} \langle t \rangle \approx u} \text{ CONFORMS_TO_REG} \\
\\
\boxed{E^D, k \vdash t_arg \approx t_arg'} \\
\\
\frac{E^D \vdash t \approx t'}{E^D, K_Typ \vdash t \approx t'} \text{ TARGCONFORMS_TYP}
\end{array}$$

$$\frac{E^D \vdash_n ne \approx ne'}{E^D, K_Nat \vdash ne \approx ne'} \quad \text{TARGCONFORMS_NEXP}$$

$$\boxed{E^D \vdash_c t \approx t'}$$

Relate t and t' when t can be used where t' is expected upto applying coercions to t

$$\frac{E^D \vdash t \approx t'}{E^D \vdash_c t \approx t'} \quad \text{CONFORMS_TO_UPTO_COERCE_BASE}$$

$$\frac{E^D \vdash_n ne_2 \approx \mathbf{one}}{E^D \vdash_c \mathbf{bit} \approx \mathbf{vector} \langle ne \ ne_2 \ order \ \mathbf{bit} \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_BITToVEC}$$

$$\frac{E^D \vdash_n ne_2 \approx \mathbf{one}}{E^D \vdash_c \mathbf{vector} \langle ne \ ne_2 \ order \ \mathbf{bit} \rangle \approx \mathbf{bit}} \quad \text{CONFORMS_TO_UPTO_COERCE_VECToBIT}$$

$$\frac{}{E^D \vdash_c \mathbf{vector} \langle ne \ ne_2 \ order \ \mathbf{bit} \rangle \approx \mathbf{atom} \langle ne_3 \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_VECToATOM}$$

$$\frac{}{E^D \vdash_c \mathbf{vector} \langle ne \ ne_2 \ order \ \mathbf{bit} \rangle \approx \mathbf{range} \langle ne_3 \ ne_4 \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_VECToRANGE}$$

$$\frac{E^E(x) \triangleright \text{enumerate_map}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c x \approx \mathbf{range} \langle ne_1 \ ne_2 \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_ENUMToRANGE}$$

$$\frac{E^E(x) \triangleright \text{enumerate_map}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c \mathbf{range} \langle ne_1 \ ne_2 \rangle \approx x} \quad \text{CONFORMS_TO_UPTO_COERCE_RANGEToENUM}$$

$$\frac{E^E(x) \triangleright \text{enumerate_map}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c x \approx \mathbf{atom} \langle ne \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_ENUMToATOM}$$

$$\frac{E^E(x) \triangleright \text{enumerate_map}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c \mathbf{atom} \langle ne \rangle \approx x} \quad \text{CONFORMS_TO_UPTO_COERCE_ATOMToENUM}$$

$$\frac{E^D \vdash_c t_1 \approx u_1 \quad \dots \quad E^D \vdash_c t_n \approx u_n}{E^D \vdash_c (t_1, \dots, t_n) \approx (u_1, \dots, u_n)} \quad \text{CONFORMS_TO_UPTO_COERCE_TUP}$$

$$\frac{E^K(x) \triangleright K_Lam(k_1 \dots k_n \rightarrow K_Typ) \quad \langle E^K, E^A, E^R, E^E \rangle, k_1 \vdash_c t_arg_1 \approx t_arg'_1 \quad \dots \quad \langle E^K, E^A, E^R, E^E \rangle, k_n \vdash_c t_arg_n \approx t_arg'_n}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c x \langle t_arg_1 \dots t_arg_n \rangle \approx x \langle t_arg'_1 \dots t_arg'_n \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_APP}$$

$$\frac{\begin{array}{l} x' \neq x \\ E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, tag, u \\ \langle E^K, E^A, E^R, E^E \rangle \vdash_c x \langle t_arg_1 \dots t_arg_n \rangle \approx u[t_arg'_1/tid_1 \dots t_arg'_m/tid_m] \end{array}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c x \langle t_arg_1 \dots t_arg_n \rangle \approx x' \langle t_arg'_1 \dots t_arg'_m \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_APPABBREV}$$

$$\frac{\begin{array}{l} x' \neq x \\ E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u \\ \langle E^K, E^A, E^R, E^E \rangle \vdash_c u[t_arg_1/tid_1 \dots t_arg_n/tid_n] \approx x \langle t_arg'_1 \dots t_arg'_m \rangle \end{array}}{\langle E^K, E^A, E^R, E^E \rangle \vdash_c x' \langle t_arg_1 \dots t_arg_n \rangle \approx x \langle t_arg'_1 \dots t_arg'_m \rangle} \quad \text{CONFORMS_TO_UPTO_COERCE_APPABBREV2}$$

$$\frac{E^D \vdash_c t \approx u}{E^D \vdash_c \mathbf{register} \langle t \rangle \approx u} \quad \text{CONFORMS_TO_UPTO_COERCE_REGISTER}$$

$$\frac{E^D \vdash_c t \approx u}{E^D \vdash_c \mathbf{reg} \langle t \rangle \approx u} \quad \text{CONFORMS_TO_UPTO_COERCE_REG}$$

$$\boxed{E^D, k \vdash_c t_arg \approx t_arg'}$$

$$\frac{E^D \vdash_c t \approx t'}{E^D, K_Typ \vdash_c t \approx t'} \quad \text{TARGCONFORMS_COERCE_TYP}$$

$$\frac{E^D \vdash_n ne \approx ne'}{E^D, K_Nat \vdash_c ne \approx ne'} \quad \text{TARGCONFORMS_COERCE_NEXP}$$

$$\boxed{\sigma_{conformsto(t,t')}(\mathit{tinflist}) \triangleright \mathit{tinflist}'}$$

$$\frac{\begin{array}{l} E^D \vdash t_i \approx t'_i \\ E^D \vdash t'_j \approx t_j \\ \sigma_{\mathbf{full}(t_i, t_j)}(\mathit{tinfl}_0 \dots \mathit{tinfl}_m \mathit{tinfl}'_0 \dots \mathit{tinfl}'_n) \triangleright \epsilon \end{array}}{\sigma_{\mathbf{full}(t_i, t_j)}(\mathit{tinfl}_0 \dots \mathit{tinfl}_m E^K, \Sigma^N, tag, t'_i \rightarrow t'_j \mathit{effect} \mathit{tinfl}'_0 \dots \mathit{tinfl}'_n) \triangleright E^K, \Sigma^N, tag, t'_i \rightarrow t'_j} \quad \text{SO_FULL}$$

$$\frac{\begin{array}{l} E^D \vdash t_i \approx t'_i \\ \sigma_{\mathbf{parm}(t_i, t_j)}(\mathit{tinfl}_0 \dots \mathit{tinfl}_m) \triangleright \epsilon \end{array}}{\sigma_{\mathbf{parm}(t_i, t_j)}(\mathit{tinfl}_0 \dots \mathit{tinfl}_m E^K, \Sigma^N, tag, t'_i \rightarrow t \mathit{effect} \mathit{tinfl}'_0 \dots \mathit{tinfl}'_n) \triangleright E^K, \Sigma^N, tag, t'_i \rightarrow t} \quad \text{SO_PARM}$$

$$\boxed{E^D, \mathit{widening} \vdash t \lesssim t' : t'', \Sigma^N}$$

t is consistent with t' if they match if t can be used where t' is needed after the constraints are solved, with no coercions needed. t'' is t

$$\begin{array}{c}
\frac{E^K \vdash_t t \text{ ok}}{\langle E^K, E^A, E^R, E^E \rangle, \text{widening} \vdash t \lesssim t : t, \{ \}} \quad \text{CONSISTENT_TYP_REFL} \\
\\
\frac{\begin{array}{c} E^D, \text{widening} \vdash t_1 \lesssim t_3 : t_4, \Sigma_1^N \\ E^D, \text{widening} \vdash t_4 \lesssim t_2 : t_5, \Sigma_2^N \end{array}}{E^D, \text{widening} \vdash t_1 \lesssim t_2 : t_5, \Sigma_1^N \uplus \Sigma_2^N} \quad \text{CONSISTENT_TYP_TRANS} \\
\\
\frac{\begin{array}{c} E^A(x) \triangleright \{ \}, \Sigma_1^N, \text{tag}, u \\ \langle E^K, E^A, E^R, E^E \rangle, \text{widening} \vdash u \lesssim t : t', \Sigma^N \end{array}}{\langle E^K, E^A, E^R, E^E \rangle, \text{widening} \vdash x \lesssim t : t', \Sigma^N \uplus \Sigma_1^N} \quad \text{CONSISTENT_TYP_ABBREV} \\
\\
\frac{\begin{array}{c} E^A(x) \triangleright \{ \}, \Sigma_1^N, \text{tag}, u \\ \langle E^K, E^A, E^R, E^E \rangle, \text{widening} \vdash t \lesssim u : t', \Sigma^{N'} \end{array}}{\langle E^K, E^A, E^R, E^E \rangle, \text{widening} \vdash t \lesssim x : t', \Sigma^N \uplus \Sigma_1^N} \quad \text{CONSISTENT_TYP_ABBREV2} \\
\\
\frac{}{E^D, \text{widening} \vdash 'x \lesssim t : t, \{ \}} \quad \text{CONSISTENT_TYP_VAR} \\
\\
\frac{}{E^D, \text{widening} \vdash t \lesssim 'x : t, \{ \}} \quad \text{CONSISTENT_TYP_VAR2} \\
\\
\frac{E^D, \text{widening} \vdash t_1 \lesssim u_1 : u'_1, \Sigma_1^N \quad \dots \quad E^D, \text{widening} \vdash t_n \lesssim u_n : u'_n, \Sigma_n^N}{E^D, \text{widening} \vdash (t_1, \dots, t_n) \lesssim (u_1, \dots, u_n) : (u'_1, \dots, u'_n), \Sigma_1^N \uplus \dots \uplus \Sigma_n^N} \quad \text{CONSISTENT_TYP_TUP} \\
\\
\frac{}{E^D, \text{widening} \vdash \mathbf{range} \langle ne_1 \ ne_2 \rangle \lesssim \mathbf{range} \langle ne_3 \ ne_4 \rangle : \mathbf{range} \langle ne_3 \ ne_4 \rangle, \{ ne_3 \leq ne_1, ne_2 \leq ne_4 \}} \quad \text{CONSISTENT_TYP_RANGE} \\
\\
\frac{}{E^D, (\mathbf{nums}, -) \vdash \mathbf{atom} \langle ne \rangle \lesssim \mathbf{range} \langle ne_1 \ ne_2 \rangle : \mathbf{atom} \langle ne \rangle, \{ ne_1 \leq ne, ne \leq ne_2 \}} \quad \text{CONSISTENT_TYP_ATOMRANGE} \\
\\
\frac{}{E^D, (\mathbf{none}, -) \vdash \mathbf{atom} \langle ne_1 \rangle \lesssim \mathbf{atom} \langle ne_2 \rangle : \mathbf{atom} \langle ne_2 \rangle, \{ ne_1 = ne_2 \}} \quad \text{CONSISTENT_TYP_ATOM} \\
\\
\frac{num_1 < num_2}{E^D, (\mathbf{nums}, -) \vdash \mathbf{atom} \langle num_1 \rangle \lesssim \mathbf{atom} \langle num_2 \rangle : \mathbf{range} \langle num_1 \ num_2 \rangle, \{ \}} \quad \text{CONSISTENT_TYP_ATOMWIDENCONST} \\
\\
\frac{num_2 < num_1}{E^D, (\mathbf{nums}, -) \vdash \mathbf{atom} \langle num_1 \rangle \lesssim \mathbf{atom} \langle num_2 \rangle : \mathbf{range} \langle num_2 \ num_1 \rangle, \{ \}} \quad \text{CONSISTENT_TYP_ATOMWIDENCONST2} \\
\\
\frac{}{E^D, \text{widening} \vdash \mathbf{range} \langle ne_1 \ ne_2 \rangle \lesssim \mathbf{atom} \langle 'x \rangle : \mathbf{atom} \langle 'x \rangle, \{ ne_1 \leq 'x, 'x \leq ne_2 \}} \quad \text{CONSISTENT_TYP_RANGEATOM}
\end{array}$$

$$\begin{array}{c}
\frac{E^D, (\mathbf{nums}, \mathbf{none}) \vdash t \lesssim t' : t'', \Sigma^N}{E^D, (-, \mathbf{none}) \vdash \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle \lesssim \mathbf{vector} \langle ne_3 \ ne_4 \ order \ t' \rangle : \mathbf{vector} \langle ne_3 \ ne_4 \ order \ t'' \rangle, \{ne_2 = ne_4, ne_1 = ne_3\} \uplus \Sigma^N} \text{CONSISTENT_TYP_VECTOR} \\
\\
\frac{E^D, (\mathbf{nums}, \mathbf{none}) \vdash t \lesssim t' : t'', \Sigma^N}{E^D, (-, \mathbf{vectors}) \vdash \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle \lesssim \mathbf{vector} \langle ne_3 \ ne_4 \ order \ t' \rangle : \mathbf{vector} \langle ne_3 \ ne_4 \ order \ t'' \rangle, \{ne_2 = ne_4\} \uplus \Sigma^N} \text{CONSISTENT_TYP_VECTORWIDEN} \\
\\
\frac{\begin{array}{c} E^K(x) \triangleright K_Lam(k_1 \dots k_n \rightarrow K_Typ) \\ \langle E^K, E^A, E^R, E^E \rangle, widening, k_1 \vdash t_arg_1 \lesssim t_arg'_1, \Sigma^N_1 \quad \dots \quad \langle E^K, E^A, E^R, E^E \rangle, widening, k_n \vdash t_arg_n \lesssim t_arg'_n, \Sigma^N_n \end{array}}{\langle E^K, E^A, E^R, E^E \rangle, widening \vdash x \langle t_arg_1 \dots t_arg_n \rangle \lesssim x \langle t_arg'_1 \dots t_arg'_n \rangle : x \langle t_arg'_1 \dots t_arg'_n \rangle, \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n} \text{CONSISTENT_TYP_APP} \\
\\
\frac{\begin{array}{c} x' \neq x \\ E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, tag, u \\ \langle E^K, E^A, E^R, E^E \rangle, widening \vdash x \langle t_arg_1 \dots t_arg_n \rangle \lesssim u[t_arg'_1/tid_1 \dots t_arg'_m/tid_m] : t, \Sigma^N_2 \end{array}}{\langle E^K, E^A, E^R, E^E \rangle, widening \vdash x \langle t_arg_1 \dots t_arg_n \rangle \lesssim x' \langle t_arg'_1 \dots t_arg'_m \rangle : x' \langle t_arg'_1 \dots t_arg'_m \rangle, \Sigma^N \uplus \Sigma^N_2} \text{CONSISTENT_TYP_APPABBREV} \\
\\
\frac{\begin{array}{c} x' \neq x \\ E^A(x') \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, tag, u \\ \langle E^K, E^A, E^R, E^E \rangle, widening \vdash u[t_arg'_1/tid_1 \dots t_arg'_m/tid_m] \lesssim x \langle t_arg_1 \dots t_arg_n \rangle : t, \Sigma^N_2 \end{array}}{\langle E^K, E^A, E^R, E^E \rangle, widening \vdash x' \langle t_arg'_1 \dots t_arg'_m \rangle \lesssim x \langle t_arg_1 \dots t_arg_n \rangle : x \langle t_arg_1 \dots t_arg_n \rangle, \Sigma^N \uplus \Sigma^N_2} \text{CONSISTENT_TYP_APPABBREV2} \\
\\
\boxed{E^D, widening, k \vdash t_arg \lesssim t_arg', \Sigma^N} \\
\\
\frac{E^D, widening \vdash t \lesssim t' : t'', \Sigma^N}{E^D, widening, K_Typ \vdash t \lesssim t', \Sigma^N} \text{TARG_CONSISTENT_TYP} \\
\\
\frac{}{E^D, widening, K_Nat \vdash ne \lesssim ne', \{ne = ne'\}} \text{TARG_CONSISTENT_NEXP} \\
\\
\boxed{E^D, widening, t' \vdash exp : t \triangleright t'', exp', \Sigma^N, effect} \\
\\
\frac{\begin{array}{c} E^D, widening, u_1 \vdash id_1 : t_1 \triangleright u_1, exp_1, \Sigma^N_1, effect_1 \quad \dots \quad E^D, widening, u_n \vdash id_n : t_n \triangleright u_n, exp_n, \Sigma^N_n, effect_n \\ exp' \equiv \mathbf{switch} \ exp \{ \mathbf{case} \ (id_1, \dots, id_n) \rightarrow (exp_1, \dots, exp_n) \} \end{array}}{E^D, widening, (u_1, \dots, u_n) \vdash exp : (t_1, \dots, t_n) \triangleright (u_1, \dots, u_n), exp', \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n, \mathbf{pure}} \text{COERCE_TYP_TUPLE} \\
\\
\frac{\begin{array}{c} E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u \lesssim t : t', \Sigma^N \\ exp' \equiv (annot)exp \end{array}}{E^D, widening, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle \vdash exp : \mathbf{vector} \langle ne_3 \ ne_4 \ order \ t' \rangle, exp', \Sigma^N \uplus \{ne_2 = ne_4\}, \mathbf{pure}} \text{COERCE_TYP_VECTORUPDATESTART}
\end{array}$$

$$\begin{array}{c}
\frac{E^D, (\mathbf{none}, \mathbf{none}) \vdash u \lesssim \mathbf{bit} : \mathbf{bit}, \Sigma^N}{exp' \equiv to_num\ exp} \\
\frac{E^D, widening, \mathbf{range} \langle ne_1\ ne_2 \rangle \vdash exp : \mathbf{vector} \langle ne_3\ ne_4\ order\ u \rangle \triangleright \mathbf{range} \langle ne_1\ ne_2 \rangle, exp', \Sigma^N \uplus \{ne_1 = \mathbf{zero}, ne_2 \geq 2 ** ne_4\}, \mathbf{pure}}{exp' \equiv to_vec\ exp} \text{COERCE_TYP_TONUM} \\
\frac{E^D, widening, \mathbf{vector} \langle ne_1\ ne_2\ order\ \mathbf{bit} \rangle \vdash exp : \mathbf{range} \langle ne_3\ ne_4 \rangle \triangleright \mathbf{vector} \langle ne_1\ ne_2\ order\ \mathbf{bit} \rangle, exp', \{ne_3 = \mathbf{zero}, ne_4 \leq 2 ** ne_2\}, \mathbf{pure}}{exp' \equiv msb\ (exp)} \text{COERCE_TYP_FROMNUM} \\
\frac{E^D \vdash typ \rightsquigarrow t \quad exp' \equiv (typ)exp \quad E^D, widening, u \vdash exp' : t \triangleright t', exp'', \Sigma^N, \mathbf{pure}}{E^D, widening, u \vdash exp : \mathbf{register} \langle t \rangle \triangleright t', exp'', \Sigma^N, \{\mathbf{rreg}\}} \text{COERCE_TYP_READREG} \\
\frac{exp' \equiv \mathbf{msb}\ (exp)}{E^D, widening, \mathbf{bit} \vdash exp : \mathbf{vector} \langle ne_1\ ne_2\ order\ \mathbf{bit} \rangle \triangleright \mathbf{bit}, exp', \{ne_1 = \mathbf{one}\}, \mathbf{pure}} \text{COERCE_TYP_ACCESSVECBit} \\
\frac{E^D, widening \vdash \mathbf{range} \langle \mathbf{zero}\ \mathbf{one} \rangle \lesssim \mathbf{range} \langle ne_1\ ne_2 \rangle : t, \Sigma^N \quad exp' \equiv \mathbf{switch}\ exp\{\mathbf{case}\ \mathbf{bitzero} \rightarrow numZero\ \mathbf{case}\ \mathbf{bitone} \rightarrow numOne\}}{E^D, widening, \mathbf{range} \langle ne_1\ ne_2 \rangle \vdash exp : \mathbf{bit} \triangleright \mathbf{range} \langle ne_1\ ne_2 \rangle, exp', \Sigma^N, \mathbf{pure}} \text{COERCE_TYP_BITTONUM} \\
\frac{E^D, widening \vdash \mathbf{range} \langle ne_1\ ne_2 \rangle \lesssim \mathbf{range} \langle \mathbf{zero}\ \mathbf{one} \rangle : t, \Sigma^N \quad exp' \equiv \mathbf{switch}\ exp\{\mathbf{case}\ numZero \rightarrow \mathbf{bitzero}\ \mathbf{case}\ numOne \rightarrow \mathbf{bitone}\}}{E^D, widening, \mathbf{bit} \vdash exp : \mathbf{range} \langle ne_1\ ne_2 \rangle \triangleright \mathbf{bit}, exp', \Sigma^N, \mathbf{pure}} \text{COERCE_TYP_NUMTOBIT} \\
\frac{E^D, (\mathbf{nums}, \mathbf{none}) \vdash \mathbf{atom} \langle ne \rangle \lesssim \mathbf{range} \langle \mathbf{zero}\ \mathbf{one} \rangle : t, \Sigma^N \quad exp' \equiv \mathbf{switch}\ exp\{\mathbf{case}\ numZero \rightarrow \mathbf{bitzero}\ \mathbf{case}\ numOne \rightarrow \mathbf{bitone}\}}{E^D, widening, \mathbf{bit} \vdash exp : \mathbf{atom} \langle ne \rangle \triangleright \mathbf{bit}, exp', \Sigma^N, \mathbf{pure}} \text{COERCE_TYP_NUMTOBITATOM} \\
\frac{E^E(x) \triangleright \{\overline{num_i} \mapsto \overline{id_i}^i\} \quad exp' \equiv \mathbf{switch}\ exp\{\mathbf{case}\ num_i \rightarrow \overline{id_i}^i\} \quad ne_3 \equiv \mathbf{count}\ (\overline{num_i}^i)}{\langle E^K, E^A, E^R, E^E \rangle, widening, x \vdash exp : \mathbf{range} \langle ne_1\ ne_2 \rangle \triangleright x, exp', \{ne_1 \leq \mathbf{zero}, ne_2 \leq ne_3\}, \mathbf{pure}} \text{COERCE_TYP_TOENUMERATE} \\
\frac{E^E(x) \triangleright \{\overline{num_i} \mapsto \overline{id_i}^i\} \quad exp' \equiv \mathbf{switch}\ exp\{\mathbf{case}\ id_i \rightarrow \overline{num_i}^i\} \quad ne_3 \equiv \mathbf{count}\ (\overline{num_i}^i) \quad \langle E^K, E^A, E^R, E^E \rangle, (\mathbf{nums}, \mathbf{none}) \vdash \mathbf{range} \langle \mathbf{zero}\ ne_3 \rangle \lesssim \mathbf{range} \langle ne_1\ ne_2 \rangle : t, \Sigma^N}{\langle E^K, E^A, E^R, E^E \rangle, widening, \mathbf{range} \langle ne_1\ ne_2 \rangle \vdash exp : x \triangleright \mathbf{range} \langle \mathbf{zero}\ ne_3 \rangle, exp', \Sigma^N, \mathbf{pure}} \text{COERCE_TYP_FROMENUMERATE}
\end{array}$$

$$\frac{E^D, \text{widening} \vdash t \lesssim u : u', \Sigma^N}{E^D, \text{widening}, u \vdash \text{exp} : t \triangleright u', \text{exp}, \Sigma^N, \mathbf{pure}} \text{COERCE_TYP_EQ}$$

$\boxed{\text{widening}, t \vdash \text{lit} : t' \Rightarrow \text{exp}, \Sigma^N}$ Typing literal constants, coercing to expected type t

$$\frac{}{\text{widening}, \mathbf{range} \langle ne \ ne' \rangle \vdash \text{num} : \mathbf{atom} \langle \text{num} \rangle \Rightarrow \text{num}, \{ne \leq \text{num}, \text{num} \leq ne'\}} \text{CHECK_LIT_NUM}$$

$$\frac{}{\text{widening}, \mathbf{vector} \langle ne \ ne' \ \text{order} \ \mathbf{bit} \rangle \vdash \text{num} : \mathbf{atom} \langle \text{num} \rangle \Rightarrow \text{to_vec} \ \text{num}, \{\text{num} + \mathbf{one} \leq 2 ** ne'\}} \text{CHECK_LIT_NUMTOVEC}$$

$$\frac{}{\text{widening}, \mathbf{bit} \vdash \text{numZero} : \mathbf{atom} \langle \mathbf{zero} \rangle \Rightarrow \mathbf{bitzero}, \{ \}} \text{CHECK_LIT_NUMBITZERO}$$

$$\frac{}{\text{widening}, \mathbf{bit} \vdash \text{numOne} : \mathbf{atom} \langle \mathbf{one} \rangle \Rightarrow \mathbf{bitone}, \{ \}} \text{CHECK_LIT_NUMBITONE}$$

$$\frac{}{\text{widening}, \text{string} \vdash \text{string} : \text{string} \Rightarrow \text{string}, \{ \}} \text{CHECK_LIT_STRING}$$

$$\frac{ne \equiv \mathbf{bitlength} \ (\text{hex})}{\text{widening}, \mathbf{vector} \langle ne_1 \ ne_2 \ \text{order} \ \mathbf{bit} \rangle \vdash \text{hex} : \mathbf{vector} \langle ne_1 \ ne \ \text{order} \ \mathbf{bit} \rangle \Rightarrow \text{hex}, \{ne = ne_2\}} \text{CHECK_LIT_HEX}$$

$$\frac{ne \equiv \mathbf{bitlength} \ (\text{bin})}{\text{widening}, \mathbf{vector} \langle ne_1 \ ne_2 \ \text{order} \ \mathbf{bit} \rangle \vdash \text{bin} : \mathbf{vector} \langle ne_1 \ ne \ \text{order} \ \mathbf{bit} \rangle \Rightarrow \text{bin}, \{ne = ne_2\}} \text{CHECK_LIT_BIN}$$

$$\frac{}{\text{widening}, \mathbf{unit} \vdash () : \mathbf{unit} \Rightarrow \mathbf{unit}, \{ \}} \text{CHECK_LIT_UNIT}$$

$$\frac{}{\text{widening}, \mathbf{bit} \vdash \mathbf{bitzero} : \mathbf{bit} \Rightarrow \mathbf{bitzero}, \{ \}} \text{CHECK_LIT_BITZERO}$$

$$\frac{}{\text{widening}, \mathbf{bit} \vdash \mathbf{bitone} : \mathbf{bit} \Rightarrow \mathbf{bitzero}, \{ \}} \text{CHECK_LIT_BITONE}$$

$$\frac{}{\text{widening}, t \vdash \mathbf{undefined} : t \Rightarrow \mathbf{undefined}, \{ \}} \text{CHECK_LIT_UNDEF}$$

$\boxed{E, t \vdash \text{pat} : t' \triangleright \text{pat}', E^T, \Sigma^N}$ Typing patterns, building their binding environment

$$\frac{\begin{array}{l} \text{lit} \neq \mathbf{undefined} \\ (\mathbf{none}, \mathbf{none}), t \vdash \text{lit} : u \Rightarrow \text{lit}', \Sigma^N \\ E^D, (\mathbf{nums}, \mathbf{none}) \vdash u \lesssim t : t', \Sigma^{N'} \end{array}}{\langle E^T, E^D \rangle, t \vdash \text{lit} : t' \triangleright \text{lit}', \{ \}, \Sigma^N \uplus \Sigma^{N'}} \text{CHECK_PAT_LIT}$$

$$\begin{array}{c}
\frac{}{E, t \vdash _ : t \triangleright _, \{\}, \{\}} \quad \text{CHECK_PAT_WILD} \\
\\
\frac{E, t \vdash pat : u \triangleright pat', E^T_1, \Sigma^N \quad id \notin \mathbf{dom}(E^T_1)}{E, t \vdash (pat \mathbf{as} id) : u \triangleright (pat' \mathbf{as} id), (E^T_1 \uplus \{id \mapsto t\}), \Sigma^N} \quad \text{CHECK_PAT_AS} \\
\\
\frac{E^T(id) \triangleright \{\}, \{\}, \mathbf{Default}, t' \quad \langle E^T, E^D \rangle, t' \vdash pat : t \triangleright pat', E^T_1, \Sigma^N \quad E^D, (\mathbf{none}, \mathbf{none}) \vdash t' \lesssim u : u', \Sigma^{N'}}{\langle E^T, E^D \rangle, u \vdash (pat \mathbf{as} id) : t \triangleright (pat' \mathbf{as} id), (E^T_1 \uplus \{id \mapsto t'\}), \Sigma^N \uplus \Sigma^{N'}} \quad \text{CHECK_PAT_ASDefault} \\
\\
\frac{E^D \vdash typ \rightsquigarrow t \quad \langle E^T, E^D \rangle, t \vdash pat : t \triangleright pat', E^T_1, \Sigma^N}{\langle E^T, E^D \rangle, u \vdash (typ)pat : t \triangleright pat', E^T_1, \Sigma^N} \quad \text{CHECK_PAT_TYP} \\
\\
\frac{E^T(id) \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, \mathbf{Ctor}, (u'_1, \dots, u'_n) \rightarrow x \langle t_arg_1 \dots t_arg_m \rangle \mathbf{pure} \quad (u_1, \dots, u_n) \rightarrow x \langle t_args' \rangle \mathbf{pure} \equiv (u'_1, \dots, u'_n) \rightarrow x \langle t_args \rangle \mathbf{pure}[t_arg_1/tid_1 \dots t_arg_m/tid_m] \quad \langle E^T, E^D \rangle, u_1 \vdash pat_1 : t_1 \triangleright pat'_1, E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, u_n \vdash pat_n : t_n \triangleright pat'_n, E^T_n, \Sigma^N_n \quad \mathbf{disjoint doms}(E^T_1, \dots, E^T_n) \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash x \langle t_args' \rangle \lesssim t : t', \Sigma^N}{\langle E^T, E^D \rangle, t \vdash id(pat_1, \dots, pat_n) : x \langle t_args' \rangle \triangleright id(pat'_1, \dots, pat'_n), \uplus E^T_1 \dots E^T_n, \Sigma^N \uplus \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n} \quad \text{CHECK_PAT_CONSTR} \\
\\
\frac{E^T(id) \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_m \mapsto kinf_m\}, \Sigma^N, \mathbf{Ctor}, \mathbf{unit} \rightarrow x \langle t_arg_1 \dots t_arg_m \rangle \mathbf{pure} \quad \mathbf{unit} \rightarrow x \langle t_args' \rangle \mathbf{pure} \equiv \mathbf{unit} \rightarrow x \langle t_args \rangle \mathbf{pure}[t_arg_1/tid_1 \dots t_arg_m/tid_m] \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash x \langle t_args' \rangle \lesssim t : t', \Sigma^N}{\langle E^T, E^D \rangle, t \vdash id : t \triangleright id, \{\}, \Sigma^N} \quad \text{CHECK_PAT_IDENTCONSTR} \\
\\
\frac{E^T(id) \triangleright \{\}, \{\}, \mathbf{Default}, t \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash t \lesssim u : u', \Sigma^N}{\langle E^T, E^D \rangle, u \vdash id : t \triangleright id, (E^T \uplus \{id \mapsto t\}), \Sigma^N} \quad \text{CHECK_PAT_VARDefault} \\
\\
\frac{}{\langle E^T, E^D \rangle, t \vdash id : t \triangleright id, (E^T \uplus \{id \mapsto t\}), \{\}} \quad \text{CHECK_PAT_VAR}
\end{array}$$

$$\begin{array}{c}
\frac{
\frac{
\frac{
E^R(\overline{id_i}^i) \triangleright x\langle t_args \rangle, (\overline{t_i}^i)
}{\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t_i \vdash pat_i : u_i \triangleright pat'_i, E^T_i, \Sigma^N_i}
}{\mathbf{disjoint\ doms}(\overline{E^T_i}^i)}
}{\langle E^K, E^A, E^R, E^E \rangle, (\mathbf{nums}, \mathbf{vectors}) \vdash x\langle t_args \rangle \lesssim t : t', \Sigma^N}
\quad \text{CHECK_PAT_RECORD} \\
\hline
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t \vdash \{ \overline{id_i} = \overline{pat_i}^i ; ? \} : x\langle t_args \rangle \triangleright \{ id_i = pat'_i{}^i ; ? \}, \uplus \overline{E^T_i}^i, \Sigma^N \uplus \overline{\Sigma^N_i}^i \\
\langle E^T, E^D \rangle, t \vdash pat_1 : u_1 \triangleright pat'_1, E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, t \vdash pat_n : u_n \triangleright pat'_n, E^T_n, \Sigma^N_n \\
\mathbf{disjoint\ doms}(E^T_1, \dots, E^T_n) \\
E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_1 \lesssim t : t', \Sigma^{N'}_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_n \lesssim t : t', \Sigma^{N'}_n \\
ne_4 \equiv \mathbf{length}(pat_1 \dots pat_n) \\
\Sigma^N \equiv \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n \\
\Sigma^{N'} \equiv \Sigma^{N'}_1 \uplus \dots \uplus \Sigma^{N'}_n \\
\hline
\langle E^T, E^D \rangle, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle \vdash [pat_1, \dots, pat_n] : \mathbf{vector} \langle ne_3 \ ne_4 \ order \ u \rangle \triangleright [pat'_1, \dots, pat'_n], (E^T_1 \uplus \dots \uplus E^T_n), \Sigma^N \uplus \Sigma^{N'} \uplus \{ ne_2 = ne_4 \}
\quad \text{CHECK_PAT_VECTOR} \\
\hline
\langle E^T, E^D \rangle, t \vdash pat_1 : u_1 \triangleright pat'_1, E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, t \vdash pat_n : u_n \triangleright pat'_n, E^T_n, \Sigma^N_n \\
E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_1 \lesssim t : t', \Sigma^{N'}_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_n \lesssim t : t', \Sigma^{N'}_n \\
ne_4 \equiv \mathbf{length}(pat_1 \dots pat_n) \\
\mathbf{disjoint\ doms}(E^T_1, \dots, E^T_n) \\
num_1 < \dots < num_n \\
\Sigma^N \equiv \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n \\
\Sigma^{N'} \equiv \Sigma^{N'}_1 \uplus \dots \uplus \Sigma^{N'}_n \\
\hline
\langle E^T, E^D \rangle, \mathbf{vector} \langle ne_1 \ ne_2 \ inc \ t \rangle \vdash [num_1 = pat_1, \dots, num_n = pat_n] : \mathbf{vector} \langle ne_3 \ ne_4 \ inc \ t \rangle \triangleright [num_1 = pat'_1, \dots, num_n = pat'_n], (E^T_1 \uplus \dots \uplus E^T_n), \{ ne_1 \leq num_1, ne_2 \geq ne_4 \} \uplus \dots \\
\hline
\langle E^T, E^D \rangle, t \vdash pat_1 : u_1 \triangleright pat'_1, E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, t \vdash pat_n : u_n \triangleright pat'_n, E^T_n, \Sigma^N_n \\
E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_1 \lesssim t : t', \Sigma^{N'}_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_n \lesssim t : t', \Sigma^{N'}_n \\
ne_4 \equiv \mathbf{length}(pat_1 \dots pat_n) \\
\mathbf{disjoint\ doms}(E^T_1, \dots, E^T_n) \\
num_1 > \dots > num_n \\
\Sigma^N \equiv \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n \\
\Sigma^{N'} \equiv \Sigma^{N'}_1 \uplus \dots \uplus \Sigma^{N'}_n \\
\hline
\langle E^T, E^D \rangle, \mathbf{vector} \langle ne_1 \ ne_2 \ dec \ t \rangle \vdash [num_1 = pat_1, \dots, num_n = pat_n] : \mathbf{vector} \langle ne_3 \ ne_4 \ dec \ t \rangle \triangleright [num_1 = pat'_1, \dots, num_n = pat'_n], (E^T_1 \uplus \dots \uplus E^T_n), \{ ne_1 \geq num_1, ne_2 \geq ne_4 \} \uplus \dots
\end{array}$$

$$\langle E^T, E^D \rangle, \mathbf{vector} \langle ne_1'' ne_1''' \text{ order } t \rangle \vdash pat_1 : \mathbf{vector} \langle ne_1'' ne_1' \text{ order } u_1 \rangle \triangleright pat_1', E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, \mathbf{vector} \langle ne_n'' ne_n''' \text{ order } t \rangle \vdash pat_n : \mathbf{vector} \langle ne_n'' ne_n' \text{ order } u_1 \rangle \triangleright pat_n', E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_1 \lesssim t : t', \Sigma^{N'}_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{vectors}) \vdash u_n \lesssim t : t', \Sigma^{N'}_n$$

disjoint doms (E^T_1, \dots, E^T_n)

$$\Sigma^N \equiv \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n$$

$$\Sigma^{N'} \equiv \Sigma^{N'}_1 \uplus \dots \uplus \Sigma^{N'}_n$$

$$\langle E^T, E^D \rangle, \mathbf{vector} \langle ne_1 ne_2 \text{ order } t \rangle \vdash pat_1 : \dots : pat_n : \mathbf{vector} \langle ne_1 ne_4 \text{ order } t \rangle \triangleright pat_1' : \dots : pat_n', (E^T_1 \uplus \dots \uplus E^T_n), \{ne_1' + \dots + ne_n' \leq ne_2\} \uplus \Sigma^N \uplus \Sigma^{N'}$$

$$E, t_1 \vdash pat_1 : u_1 \triangleright pat_1', E^T_1, \Sigma^N_1 \quad \dots \quad E, t_n \vdash pat_n : u_n \triangleright pat_n', E^T_n, \Sigma^N_n$$

disjoint doms (E^T_1, \dots, E^T_n)

$$E, (t_1, \dots, t_n) \vdash (pat_1, \dots, pat_n) : (u_1, \dots, u_n) \triangleright (pat_1', \dots, pat_n'), (E^T_1 \uplus \dots \uplus E^T_n), \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n \quad \text{CHECK_PAT_TUP}$$

$$\langle E^T, E^D \rangle, t \vdash pat_1 : u_1 \triangleright pat_1', E^T_1, \Sigma^N_1 \quad \dots \quad \langle E^T, E^D \rangle, t \vdash pat_n : u_n \triangleright pat_n', E^T_n, \Sigma^N_n$$

disjoint doms (E^T_1, \dots, E^T_n)

$$E^D, (\mathbf{nums}, \mathbf{none}) \vdash u_1 \lesssim t : t', \Sigma^{N'}_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{none}) \vdash u_n \lesssim t : t', \Sigma^{N'}_n$$

disjoint doms (E^T_1, \dots, E^T_n)

$$\Sigma^N \equiv \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n$$

$$\Sigma^{N'} \equiv \Sigma^{N'}_1 \uplus \dots \uplus \Sigma^{N'}_n$$

$$\langle E^T, E^D \rangle, \mathbf{list} \langle t \rangle \vdash [||pat_1, \dots, pat_n||] : \mathbf{list} \langle t \rangle \triangleright [||pat_1', \dots, pat_n'||], (E^T_1 \uplus \dots \uplus E^T_n), \Sigma^N \uplus \Sigma^{N'} \quad \text{CHECK_PAT_LIST}$$

$$E, t, \text{widening} \vdash exp : t' \triangleright exp', I, E^T$$

Typing expressions, collecting nexp constraints, effects, and new bindings

$$E^T(id) \triangleright \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \{\}, \mathbf{Ctor}, \mathbf{unit} \rightarrow x \langle t_args \rangle \mathbf{pure}$$

$$u \equiv x \langle t_args \rangle [t_arg_0 / tid_0 \dots t_arg_n / tid_n]$$

$$E^D, \text{widening} \vdash u \lesssim t : t', \Sigma^N$$

$$\langle E^T, E^D \rangle, t, \text{widening} \vdash id : x \triangleright id, \langle \Sigma^N, \mathbf{pure} \rangle, \{\} \quad \text{CHECK_EXP_UNARYCTOR}$$

$$E^T(id) \triangleright \{\}, \{\}, tag, u$$

$$E^D, \text{widening}, t \vdash id : u \triangleright t', exp, \Sigma^N, effect$$

$$\langle E^T, E^D \rangle, t, \text{widening} \vdash id : u \triangleright id, \langle \Sigma^N, effect \rangle, \{\} \quad \text{CHECK_EXP_LOCALVAR}$$

$$E^T(id) \triangleright \{tid_1 \mapsto kinf_1, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u'$$

$$u \equiv u' [t_arg_1 / tid_1 \dots t_arg_n / tid_n]$$

$$E^D, \text{widening}, t \vdash id : u \triangleright t', exp, \Sigma^{N'}, effect$$

$$\langle E^T, E^D \rangle, t, \text{widening} \vdash id : u \triangleright id, \langle \Sigma^N \uplus \Sigma^{N'}, effect \rangle, \{\} \quad \text{CHECK_EXP_OTHERVAR}$$

$$\begin{array}{c}
\frac{
\begin{array}{l}
E^T(id) \triangleright \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \{\}, \mathbf{Ctor}, t'' \rightarrow x\langle t_args \rangle \mathbf{pure} \\
t' \rightarrow u \mathbf{pure} \equiv t'' \rightarrow x\langle t_args \rangle \mathbf{pure}[t_arg_0/tid_0 \dots t_arg_n/tid_n] \\
E^D, widening \vdash u \lesssim t : t', \Sigma^N \\
\langle E^T, E^D \rangle, t, widening \vdash exp : u' \triangleright exp, \langle \Sigma^{N'}, effect \rangle, E^{T'}
\end{array}
}{
\langle E^T, E^D \rangle, t, widening \vdash id(exp) : t \triangleright id(exp'), \langle \Sigma^N \uplus \Sigma^N, effect \rangle, \{\}
} \text{ CHECK_EXP_CTOR}
\\[10pt]
\frac{
\begin{array}{l}
E^T(id) \triangleright \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u \\
u[t_arg_0/tid_0 \dots t_arg_n/tid_n] \equiv u_i \rightarrow u_j \text{ effect} \\
u_i \equiv (\mathbf{implicit} \langle ne \rangle, t_0, \dots, t_m) \\
\langle E^T, E^D \rangle, (t_0, \dots, t_m), widening \vdash (exp_0, \dots, exp_m) : u'_i \triangleright (exp'_0, \dots, exp'_m), I, E^{T'} \\
E^D, widening, t \vdash id(annot, exp'_0, \dots, exp'_m) : u_j \triangleright u'_j, exp'', \Sigma^{N'}, effect'
\end{array}
}{
\langle E^T, E^D \rangle, t, widening \vdash id(exp_0, \dots, exp_m) : u_j \triangleright exp'', I \uplus \langle \Sigma^N, effect \rangle \uplus \langle \Sigma^{N'}, effect' \rangle, E^T
} \text{ CHECK_EXP_APPIMPLICIT}
\\[10pt]
\frac{
\begin{array}{l}
E^T(id) \triangleright \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u \\
u[t_arg_0/tid_0 \dots t_arg_n/tid_n] \equiv u_i \rightarrow u_j \text{ effect} \\
\langle E^T, E^D \rangle, u_i, widening \vdash exp : u'_i \triangleright exp', I, E^{T'} \\
E^D, widening, t \vdash id(exp') : u_j \triangleright u'_j, exp'', \Sigma^{N'}, effect'
\end{array}
}{
\langle E^T, E^D \rangle, t, widening \vdash id(exp) : u_j \triangleright exp'', I \uplus \langle \Sigma^N, effect \rangle \uplus \langle \Sigma^{N'}, effect' \rangle, E^T
} \text{ CHECK_EXP_APP}
\\[10pt]
\frac{
\begin{array}{l}
E^T(id) \triangleright \mathbf{overload} \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u : tinf_1 \dots tinf_n \\
u[t_arg_0/tid_0 \dots t_arg_n/tid_n] \equiv u_i \rightarrow u_j \text{ effect} \\
\langle E^T, E^D \rangle, u_i, widening \vdash exp : u'_i \triangleright exp', I, E^{T'} \\
\sigma_{\mathbf{full}(u'_i, t)}(tinf_1 \dots tinf_n) \triangleright tinf \\
\langle (\{id \mapsto tinf\} \uplus E^T), E^D \rangle, t, widening \vdash id(exp) : t' \triangleright exp'', I', E^{T''}
\end{array}
}{
\langle E^T, E^D \rangle, t, widening \vdash id(exp) : u_j \triangleright exp'', I \uplus I' \uplus \langle \Sigma^N, effect \rangle \uplus \langle \Sigma^{N'}, effect' \rangle, E^T
} \text{ CHECK_EXP_APPOVERLOAD}
\\[10pt]
\frac{
\begin{array}{l}
E^T(id) \triangleright \{tid_0 \mapsto kinf_0, \dots, tid_n \mapsto kinf_n\}, \Sigma^N, tag, u \\
u[t_arg_0/tid_0 \dots t_arg_n/tid_n] \equiv u_i \rightarrow u_j \text{ effect} \\
\langle E^T, E^D \rangle, u_i, widening \vdash (exp_1, exp_2) : u'_i \triangleright (exp'_1, exp'_2), I, E^{T'} \\
E^D, widening, t \vdash exp'_1 id exp'_2 : u_j \triangleright u'_j, exp, \Sigma^{N'}, effect'
\end{array}
}{
\langle E^T, E^D \rangle, t, widening \vdash exp_1 id exp_2 : t \triangleright exp, I \uplus \langle \Sigma^N, effect \rangle \uplus \langle \Sigma^{N'}, effect' \rangle, E^T
} \text{ CHECK_EXP_INFIX_APP}
\end{array}$$

$$\begin{array}{c}
\frac{
\begin{array}{l}
E^T(id) \triangleright \mathbf{overload} \{tid_0 \mapsto \mathit{kinf}_0, \dots, tid_n \mapsto \mathit{kinf}_n\}, \Sigma^N, tag, u : \mathit{tinf}_1 \dots \mathit{tinf}_n \\
u[t_arg_0/tid_0 \dots t_arg_n/tid_n] \equiv u_i \rightarrow u_j \mathit{effect} \\
\langle E^T, E^D \rangle, u_i, \mathit{widening} \vdash (exp_1, exp_2) : u'_i \triangleright (exp'_1, exp'_2), I, E^{T'} \\
\sigma_{\mathbf{full}(u'_i, t)}(\mathit{tinf}_1 \dots \mathit{tinf}_n) \triangleright \mathit{tinf} \\
\langle \{id \mapsto \mathit{tinf}\} \uplus E^T \rangle, E^D \rangle, t, \mathit{widening} \vdash exp_1 \mathit{id} exp_2 : t' \triangleright exp, I', E^{T''}
\end{array}
}{
\langle E^T, E^D \rangle, t, \mathit{widening} \vdash exp_1 \mathit{id} exp_2 : t \triangleright exp, I \uplus I \uplus \langle \Sigma^N, \mathit{effect} \rangle \uplus \langle \Sigma^{N'}, \mathit{effect}' \rangle, E^T
} \quad \text{CHECK_EXP_INFIX_APPOVERLOAD}
\\[10pt]
\frac{
\begin{array}{l}
E^R(\overline{id_i}^i) \triangleright x \langle t_args \rangle, \overline{t_i}^i \\
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t_i, \mathit{widening} \vdash exp_i : u_i \triangleright exp'_i, \langle \Sigma^N_i, \mathit{effect}_i \rangle, E^{T^i} \\
\langle E^K, E^A, E^R, E^E \rangle, \mathit{widening} \vdash u_i \lesssim t_i : t'_i, \Sigma^{N'}_i \\
\Sigma^N \equiv \uplus \overline{\Sigma^N_i}^i \\
\Sigma^{N'} \equiv \uplus \overline{\Sigma^{N'}_i}^i
\end{array}
}{
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t, \mathit{widening} \vdash \{\overline{id_i = exp_i^i}; ?\} : x \langle t_args \rangle \triangleright \{\overline{id_i = exp'_i^i}; ?\}, \uplus \langle \Sigma^N \uplus \Sigma^{N'}, \uplus \overline{effect_i^i} \rangle, \{ \}
} \quad \text{CHECK_EXP_RECORD}
\\[10pt]
\frac{
\begin{array}{l}
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t, \mathit{widening} \vdash exp : x \langle t_args \rangle \triangleright exp', I, E^T \\
E^R(x \langle t_args \rangle) \triangleright \overline{id'_n : t'_n}^n \\
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t_i, \mathit{widening} \vdash exp_i : u_i \triangleright exp'_i, I_i, E^{T^i} \\
\overline{id_i : t_i}^i \subset \overline{id'_n : t'_n}^n \\
\langle E^K, E^A, E^R, E^E \rangle, \mathit{widening} \vdash u_i \lesssim t_i : t'_i, \Sigma^{N'}_i
\end{array}
}{
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t, \mathit{widening} \vdash \{exp \mathbf{with} \overline{id_i = exp_i^i}; ?\} : x \langle t_args \rangle \triangleright \{exp' \mathbf{with} \overline{id_i = exp'_i^i}\}, I \uplus \overline{I_i}^i, E^T
} \quad \text{CHECK_EXP_RECUP}
\\[10pt]
\frac{
\begin{array}{l}
\langle E^T, E^D \rangle, t, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : u_1 \triangleright exp'_1, I_1, E^{T'} \quad \dots \quad \langle E^T, E^D \rangle, t, (\mathbf{nums}, \mathbf{none}) \vdash exp_n : u_n \triangleright exp'_n, I_n, E^{T'} \\
E^D, (\mathbf{nums}, \mathbf{none}) \vdash u_1 \lesssim t : t', \Sigma^N_1 \quad \dots \quad E^D, (\mathbf{nums}, \mathbf{none}) \vdash u_n \lesssim t : t', \Sigma^N_n \\
\mathbf{length}(exp_1 \dots exp_n) \equiv ne \\
\Sigma^N \equiv \{ne = ne_2\} \uplus \Sigma^N_1 \uplus \dots \uplus \Sigma^N_n
\end{array}
}{
E, \mathbf{vector} \langle ne_1 ne_2 \mathit{order} t \rangle, \mathit{widening} \vdash [exp_1, \dots, exp_n] : \mathbf{vector} \langle ne_1 \mathit{num} \mathit{order} t \rangle \triangleright [exp'_1, \dots, exp'_n], \langle \Sigma^N, \mathbf{pure} \rangle \uplus I_1 \uplus \dots \uplus I_n, E^T
} \quad \text{CHECK_EXP_VECTOR}
\\[10pt]
\frac{
\begin{array}{l}
E, \mathbf{vector} \langle ne ne' \mathit{order} t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : \mathbf{vector} \langle ne_1 ne'_1 \mathbf{inc} u \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{range} \langle ne_2 ne'_2 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{range} \langle ne_3 ne'_3 \rangle \triangleright exp'_2, I_2, E^T
\end{array}
}{
E, t, \mathit{widening} \vdash exp_1[exp_2] : u \triangleright exp'_1[exp'_2], I_1 \uplus I_2 \uplus \langle \{ne_1 \leq ne_3, ne_3 + ne'_3 \leq ne_1 + ne'_1\}, \mathbf{pure} \rangle, E^T
} \quad \text{CHECK_EXP_VECTORGETINC}
\end{array}$$

$$\begin{array}{c}
\frac{
\begin{array}{l}
E, \mathbf{vector} \langle ne \ ne' \ order \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : \mathbf{vector} \langle ne_1 \ ne'_1 \ dec \ u \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{range} \langle ne_2 \ ne'_2 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{range} \langle ne_3 \ ne'_3 \rangle \triangleright exp'_2, I_2, E^T
\end{array}
}{
E, t, widening \vdash exp_1[exp_2] : u \triangleright exp'_1[exp'_2], I_1 \uplus I_2 \uplus \langle \{ ne_1 \geq ne_3, ne_3 + (-ne'_3) \leq ne_1 + (-ne'_1) \}, \mathbf{pure} \rangle, E^T
} \quad \text{CHECK_EXP_VECTORGETDEC}
\\
\frac{
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 \ ne'_1 \ inc \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : \mathbf{vector} \langle ne_2 \ ne'_2 \ inc \ u \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{range} \langle ne_3 \ ne'_3 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{range} \langle ne_4 \ ne'_4 \rangle \triangleright exp'_2, I_2, E^T \\
E, \mathbf{range} \langle ne_5 \ ne'_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_3 : \mathbf{range} \langle ne_6 \ ne'_6 \rangle \triangleright exp'_3, I_3, E^T
\end{array}
}{
E, \mathbf{vector} \langle ne \ ne' \ inc \ t \rangle, widening \vdash exp_1[exp_2..exp_3] : \mathbf{vector} \langle ne_7 \ ne'_7 \ inc \ u \rangle \triangleright exp'_1[exp'_2 : exp'_3], I_1 \uplus I_2 \uplus I_3 \uplus \langle \{ ne \geq ne_4, ne \leq ne'_4, ne' \leq ne_4 + ne'_6, ne_4 \leq ne_2, ne_4 + ne'_6 \leq
\\
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 \ ne'_1 \ dec \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : \mathbf{vector} \langle ne_2 \ ne'_2 \ dec \ u \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{range} \langle ne_3 \ ne'_3 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{range} \langle ne_4 \ ne'_4 \rangle \triangleright exp'_2, I_2, E^T \\
E, \mathbf{range} \langle ne_5 \ ne'_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_3 : \mathbf{range} \langle ne_6 \ ne'_6 \rangle \triangleright exp'_3, I_3, E^T
\end{array}
\\
E, \mathbf{vector} \langle ne \ ne' \ dec \ t \rangle, widening \vdash exp_1[exp_2..exp_3] : \mathbf{vector} \langle ne_7 \ ne'_7 \ dec \ u \rangle \triangleright exp'_1[exp'_2 : exp'_3], I_1 \uplus I_2 \uplus I_3 \uplus \langle \{ ne \leq ne_4, ne \geq ne'_4, ne' \leq ne'_6 + (-ne_4), ne'_4 \geq ne_2, ne'_6 + (-
\\
\begin{array}{l}
E, \mathbf{vector} \langle ne \ ne' \ inc \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : \mathbf{vector} \langle ne_1 \ ne_2 \ inc \ u \rangle \triangleright exp', I, E^T \\
E, \mathbf{range} \langle ne'_1 \ ne'_2 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_1 : \mathbf{range} \langle ne_3 \ ne_4 \rangle \triangleright exp'_1, I_1, E^T \\
E, t, (\mathbf{nums}, \mathbf{vectors}) \vdash exp_2 : u \triangleright exp'_2, I_2, E^T
\end{array}
\\
E, \mathbf{vector} \langle ne \ ne' \ inc \ t \rangle, widening \vdash [exp \ \mathbf{with} \ exp_1 = exp_2] : \mathbf{vector} \langle ne_1 \ ne_2 \ inc \ u \rangle \triangleright [exp' \ \mathbf{with} \ exp'_1 = exp'_2], I \uplus I_1 \uplus I_2 \uplus \langle \{ ne_1 \leq ne_3, ne_2 \geq ne_4 \}, \mathbf{pure} \rangle, E^T \quad \text{CHECK_EXP}
\\
\begin{array}{l}
E, \mathbf{vector} \langle ne \ ne' \ dec \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : \mathbf{vector} \langle ne_1 \ ne_2 \ dec \ u \rangle \triangleright exp', I, E^T \\
E, \mathbf{range} \langle ne'_1 \ ne'_2 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_1 : \mathbf{range} \langle ne_3 \ ne_4 \rangle \triangleright exp'_1, I_1, E^T \\
E, t, (\mathbf{nums}, \mathbf{vectors}) \vdash exp_2 : u \triangleright exp'_2, I_2, E^T
\end{array}
\\
E, \mathbf{vector} \langle ne \ ne' \ dec \ t \rangle, widening \vdash [exp \ \mathbf{with} \ exp_1 = exp_2] : \mathbf{vector} \langle ne_1 \ ne_2 \ dec \ u \rangle \triangleright [exp' \ \mathbf{with} \ exp'_1 = exp'_2], I \uplus I_1 \uplus I_2 \uplus \langle \{ ne_1 \geq ne_3, ne_2 \geq ne_4 \}, \mathbf{pure} \rangle, E^T \quad \text{CHECK_EXI}
\\
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : \mathbf{vector} \langle ne_3 \ ne_4 \ inc \ u \rangle \triangleright exp', I, E^T \\
E, \mathbf{atom} \langle ne_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_1 : \mathbf{atom} \langle ne_6 \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{atom} \langle ne_7 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{atom} \langle ne_8 \rangle \triangleright exp'_2, I_2, E^T \\
E, \mathbf{vector} \langle ne_9 \ ne_{10} \ inc \ t \rangle, (\mathbf{nums}, \mathbf{vectors}) \vdash exp_3 : \mathbf{vector} \langle ne_{11} \ ne_{12} \ inc \ u \rangle \triangleright exp'_3, I_3, E^T \\
I_4 \equiv \langle \{ ne_3 \leq ne_5, ne_3 + ne_4 \leq ne_7, ne_{12} = ne_8 + (-ne_6), ne_6 + \mathbf{one} \leq ne_8 \}, \mathbf{pure} \rangle
\end{array}
\\
E, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle, widening \vdash [exp \ \mathbf{with} \ exp_1 : exp_2 = exp_3] : \mathbf{vector} \langle ne_3 \ ne_4 \ inc \ u \rangle \triangleright [exp' \ \mathbf{with} \ exp'_1 : exp'_2 = exp'_3], I \uplus I_1 \uplus I_2 \uplus I_3 \uplus I_4, E^T \quad \text{CHECK_EXP_VECRAN}
\\
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : \mathbf{vector} \langle ne_3 \ ne_4 \ inc \ u \rangle \triangleright exp', I, E^T \\
E, \mathbf{atom} \langle ne_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_1 : \mathbf{atom} \langle ne_6 \rangle \triangleright exp'_1, I_1, E^T \\
E, \mathbf{atom} \langle ne_7 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash exp_2 : \mathbf{atom} \langle ne_8 \rangle \triangleright exp'_2, I_2, E^T \\
E, u, (\mathbf{nums}, \mathbf{vectors}) \vdash exp_3 : u' \triangleright exp'_3, I_3, E^T \\
I_4 \equiv \langle \{ ne_3 \leq ne_5, ne_3 + ne_4 \leq ne_7 \}, \mathbf{pure} \rangle
\end{array}
\\
E, \mathbf{vector} \langle ne_1 \ ne_2 \ order \ t \rangle, widening \vdash [exp \ \mathbf{with} \ exp_1 : exp_2 = exp_3] : \mathbf{vector} \langle ne_3 \ ne_4 \ inc \ u \rangle \triangleright [exp' \ \mathbf{with} \ exp'_1 : exp'_2 = exp'_3], I \uplus I_1 \uplus I_2 \uplus I_3 \uplus I_4, E^T \quad \text{CHECK_EXP_VECRAN}
\end{array}$$

$$\begin{array}{c}
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 ne_2 \text{ order } t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash \text{exp} : \mathbf{vector} \langle ne_3 ne_4 \mathbf{dec} u \rangle \triangleright \text{exp}', I, E^T \\
E, \mathbf{atom} \langle ne_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash \text{exp}_1 : \mathbf{atom} \langle ne_6 \rangle \triangleright \text{exp}'_1, I_1, E^T \\
E, \mathbf{atom} \langle ne_7 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash \text{exp}_2 : \mathbf{atom} \langle ne_8 \rangle \triangleright \text{exp}'_2, I_2, E^T \\
E, \mathbf{vector} \langle ne_9 ne_{10} \mathbf{dec} t \rangle, (\mathbf{nums}, \mathbf{vectors}) \vdash \text{exp}_3 : \mathbf{vector} \langle ne_{11} ne_{12} \mathbf{dec} u \rangle \triangleright \text{exp}'_3, I_3, E^T \\
I_4 \equiv \langle \{ ne_5 \leq ne_3, ne_3 + (-ne_4) \leq ne_6 + (-ne_8), ne_8 + \mathbf{one} \leq ne_6 \}, \mathbf{pure} \rangle
\end{array} \\
\hline
E, \mathbf{vector} \langle ne_1 ne_2 \text{ order } t \rangle, \text{widening} \vdash [\text{exp with } \text{exp}_1 : \text{exp}_2 = \text{exp}_3] : \mathbf{vector} \langle ne_3 ne_4 \mathbf{dec} u \rangle \triangleright [\text{exp}' \text{ with } \text{exp}'_1 : \text{exp}'_2 = \text{exp}'_3], I \uplus I_1 \uplus I_2 \uplus I_3 \uplus I_4, E^T
\end{array} \quad \text{CHECK_EXP_VECRAN}$$

$$\begin{array}{c}
\begin{array}{l}
E, \mathbf{vector} \langle ne_1 ne_2 \text{ order } t \rangle, (\mathbf{nums}, \mathbf{none}) \vdash \text{exp} : \mathbf{vector} \langle ne_3 ne_4 \mathbf{dec} u \rangle \triangleright \text{exp}', I, E^T \\
E, \mathbf{atom} \langle ne_5 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash \text{exp}_1 : \mathbf{atom} \langle ne_6 \rangle \triangleright \text{exp}'_1, I_1, E^T \\
E, \mathbf{atom} \langle ne_7 \rangle, (\mathbf{none}, \mathbf{vectors}) \vdash \text{exp}_2 : \mathbf{atom} \langle ne_8 \rangle \triangleright \text{exp}'_2, I_2, E^T \\
E, u, (\mathbf{nums}, \mathbf{vectors}) \vdash \text{exp}_3 : u' \triangleright \text{exp}'_3, I_3, E^T \\
I_4 \equiv \langle \{ ne_5 \leq ne_3, ne_3 + (-ne_4) \leq ne_6 + (-ne_8), ne_8 + \mathbf{one} \leq ne_6 \}, \mathbf{pure} \rangle
\end{array} \\
\hline
E, \mathbf{vector} \langle ne_1 ne_2 \text{ order } t \rangle, \text{widening} \vdash [\text{exp with } \text{exp}_1 : \text{exp}_2 = \text{exp}_3] : \mathbf{vector} \langle ne_3 ne_4 \mathbf{dec} u \rangle \triangleright [\text{exp}' \text{ with } \text{exp}'_1 : \text{exp}'_2 = \text{exp}'_3], I \uplus I_1 \uplus I_2 \uplus I_3 \uplus I_4, E^T
\end{array} \quad \text{CHECK_EXP_VECRAN}$$

$$\begin{array}{c}
\begin{array}{l}
E^R(x \langle t_args \rangle) \triangleright \overline{id_i : t_i^i id : u id'_j : t'_j^j} \\
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t'', \text{widening} \vdash \text{exp} : x \langle t_args \rangle \triangleright \text{exp}', I, E^T \\
E^D, \text{widening}, t \vdash \text{exp}' . id : u \triangleright t', \text{exp}'_1, \Sigma^{N'}, \text{effect}
\end{array} \\
\hline
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, t, \text{widening} \vdash \text{exp} . id : u \triangleright \text{exp}'_1, I \uplus \langle \Sigma^{N'}, \text{effect} \rangle, E^T
\end{array} \quad \text{CHECK_EXP_FIELD}$$

$$\begin{array}{c}
\begin{array}{l}
\langle E^T, E^D \rangle, t'', \text{widening} \vdash \text{exp} : u \triangleright \text{exp}', I, E^T \\
\langle E^T, E^D \rangle, u \vdash \text{pat}_i : u'_i \triangleright \text{pat}'_i, E^{T_i}, \Sigma^{N_i^i} \\
\langle (E^T \uplus E^{T_i}), E^D \rangle, t, \text{widening} \vdash \text{exp}_i : u''_i \triangleright \text{exp}'_i, I_i, E^{T_i^i}
\end{array} \\
\hline
\langle E^T, E^D \rangle, t, \text{widening} \vdash \mathbf{switch} \text{exp} \{ \mathbf{case} \text{pat}_i \rightarrow \text{exp}_i^i \} : u \triangleright \mathbf{switch} \text{exp}' \{ \mathbf{case} \text{pat}'_i \rightarrow \text{exp}'_i^i \}, I \uplus I_i \uplus \langle \Sigma^{N_i}, \mathbf{pure} \rangle^i, E^T
\end{array} \quad \text{CHECK_EXP_CASE}$$

$$\begin{array}{c}
\begin{array}{l}
\langle E^T, E^D \rangle, t'', \text{widening} \vdash \text{exp} : u \triangleright \text{exp}', I, E^T \\
E^D \vdash \text{typ} \rightsquigarrow t' \\
E^D, \text{widening}, t' \vdash \text{exp}' : u \triangleright u', \text{exp}'', \Sigma^N, \text{effect} \\
E^D, \text{widening}, t \vdash \text{exp}'' : t' \triangleright u'', \text{exp}''', \Sigma^{N'}, \text{effect}'
\end{array} \\
\hline
\langle E^T, E^D \rangle, t, \text{widening} \vdash (\text{typ}) \text{exp} : t \triangleright \text{exp}''', I \uplus \langle \Sigma^N \uplus \Sigma^{N'}, \text{effect} \uplus \text{effect}' \rangle, E^T
\end{array} \quad \text{CHECK_EXP_TYPED}$$

$$\begin{array}{c}
\begin{array}{l}
\langle E^T, E^D \rangle \vdash \text{letbind} \triangleright \text{letbind}', E^{T_1}, \Sigma^N, \text{effect}, \{ \} \\
\langle (E^T \uplus E^{T_1}), E^D \rangle, t, \text{widening} \vdash \text{exp} : u \triangleright \text{exp}', I_2, E^{T_2}
\end{array} \\
\hline
\langle E^T, E^D \rangle, t, \text{widening} \vdash \text{letbind in } \text{exp} : t \triangleright \text{letbind' in } \text{exp}', \langle \Sigma^N, \text{effect} \rangle \uplus I_2, E^T
\end{array} \quad \text{CHECK_EXP_LET}$$

$$\begin{array}{c}
\frac{E, t_1, \text{widening} \vdash \text{exp}_1 : u_1 \triangleright \text{exp}'_1, I_1, E^{\text{T}}_1 \quad \dots \quad E, t_n, \text{widening} \vdash \text{exp}_n : u_n \triangleright \text{exp}'_n, I_n, E^{\text{T}}_n}{E, (t_1, \dots, t_n), \text{widening} \vdash (\text{exp}_1, \dots, \text{exp}_n) : (u_1, \dots, u_n) \triangleright (\text{exp}'_1, \dots, \text{exp}'_n), I_1 \uplus \dots \uplus I_n, E^{\text{T}}} \text{ CHECK_EXP_TUP} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, t, (\mathbf{nums}, \mathbf{none}) \vdash \text{exp}_1 : u_1 \triangleright \text{exp}'_1, I_1, E^{\text{T}}_1 \quad \dots \quad \langle E^{\text{T}}, E^{\text{D}} \rangle, t, (\mathbf{nums}, \mathbf{none}) \vdash \text{exp}_n : u_n \triangleright \text{exp}'_n, I_n, E^{\text{T}}_n}{E^{\text{D}}, (\mathbf{nums}, \mathbf{none}) \vdash u_1 \lesssim t : t', \Sigma^{\text{N}}_1 \quad \dots \quad E^{\text{D}}, (\mathbf{nums}, \mathbf{none}) \vdash u_n \lesssim t : t', \Sigma^{\text{N}}_n} \text{ CHECK_EXP_LIST} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{list} \langle t \rangle, \text{widening} \vdash [|\text{exp}_1, \dots, \text{exp}_n|] : \mathbf{list} \langle u \rangle \triangleright [|\text{exp}'_1, \dots, \text{exp}'_n|], \langle \Sigma^{\text{N}}_1 \uplus \dots \uplus \Sigma^{\text{N}}_n, \mathbf{pure} \rangle \uplus I_1 \uplus \dots \uplus I_n, E^{\text{T}}}{\begin{array}{l} E, \mathbf{bit}, \text{widening} \vdash \text{exp}_1 : \mathbf{bit} \triangleright \text{exp}'_1, I_1, E^{\text{T}'} \\ E, t, \text{widening} \vdash \text{exp}_2 : u_1 \triangleright \text{exp}'_2, I_2, E^{\text{T}}_2 \\ E, t, \text{widening} \vdash \text{exp}_3 : u_2 \triangleright \text{exp}'_3, I_3, E^{\text{T}}_3 \\ E^{\text{D}}, \text{widening} \vdash u_1 \lesssim t : t', \Sigma^{\text{N}}_1 \\ E^{\text{D}}, \text{widening} \vdash u_2 \lesssim t : t', \Sigma^{\text{N}}_2 \end{array}} \text{ CHECK_EXP_IF} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, t, \text{widening} \vdash \mathbf{if} \text{exp}_1 \mathbf{then} \text{exp}_2 \mathbf{else} \text{exp}_3 : u \triangleright \mathbf{if} \text{exp}'_1 \mathbf{then} \text{exp}'_2 \mathbf{else} \text{exp}'_3, \langle \Sigma^{\text{N}}_1 \uplus \Sigma^{\text{N}}_2, \mathbf{pure} \rangle \uplus I_1 \uplus I_2 \uplus I_3, (E^{\text{T}}_2 \cap E^{\text{T}}_3)}{\begin{array}{l} \langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{range} \langle ne_1 ne_2 \rangle, \text{widening} \vdash \text{exp}_1 : \mathbf{range} \langle ne_7 ne_8 \rangle \triangleright \text{exp}'_1, I_1, E^{\text{T}} \\ \langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{range} \langle ne_3 ne_4 \rangle, \text{widening} \vdash \text{exp}_2 : \mathbf{range} \langle ne_9 ne_{10} \rangle \triangleright \text{exp}'_2, I_2, E^{\text{T}} \\ \langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{range} \langle ne_5 ne_6 \rangle, \text{widening} \vdash \text{exp}_3 : \mathbf{range} \langle ne_{11} ne_{12} \rangle \triangleright \text{exp}'_3, I_3, E^{\text{T}} \\ \langle (E^{\text{T}} \uplus \{id \mapsto \mathbf{range} \langle ne_1 ne_4 \rangle\}), E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \text{exp}_4 : t \triangleright \text{exp}'_4, I_4, E^{\text{T}'} \end{array}} \text{ CHECK_EXP_CONS} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \mathbf{foreach} (id \mathbf{from} \text{exp}_1 \mathbf{to} \text{exp}_2 \mathbf{by} \text{exp}_3) \text{exp}_4 : t \triangleright \mathbf{foreach} (id \mathbf{from} \text{exp}'_1 \mathbf{to} \text{exp}'_2 \mathbf{by} \text{exp}'_3) \text{exp}'_4, I_1 \uplus I_2 \uplus I_3 \uplus I_4 \uplus \langle \{ne_1 \leq ne_3 + ne_4\}, \mathbf{pure} \rangle, E^{\text{T}}}{\begin{array}{l} E, t, \text{widening} \vdash \text{exp}_1 : u \triangleright \text{exp}'_1, I_1, E^{\text{T}} \\ E, \mathbf{list} \langle t \rangle, \text{widening} \vdash \text{exp}_2 : \mathbf{list} \langle u \rangle \triangleright \text{exp}'_2, I_2, E^{\text{T}} \end{array}} \text{ CHECK_EXP_CONS} \\
\\
\frac{\text{widening}, t \vdash lit : u \Rightarrow \text{exp}, \Sigma^{\text{N}}}{E, t, \text{widening} \vdash lit : u \triangleright \text{exp}, \langle \Sigma^{\text{N}}, \mathbf{pure} \rangle, E^{\text{T}}} \text{ CHECK_EXP_LIT} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \text{exp} : \mathbf{unit} \triangleright \text{exp}', I, E^{\text{T}}_1}{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \{\text{exp}\} : \mathbf{unit} \triangleright \{\text{exp}'\}, I, E^{\text{T}}} \text{ CHECK_EXP_BLOCKBASE} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \text{exp} : \mathbf{unit} \triangleright \text{exp}', I_1, E^{\text{T}}_1}{\langle (E^{\text{T}} \uplus E^{\text{T}}_1), E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \{\overline{\text{exp}}_i^i\} : \mathbf{unit} \triangleright \{\overline{\text{exp}}_i^i\}, I_2, E^{\text{T}}_2} \text{ CHECK_EXP_BLOCKREC} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \{\text{exp}; \overline{\text{exp}}_i^i\} : \mathbf{unit} \triangleright \{\text{exp}'; \overline{\text{exp}}_i^i\}, I_1 \uplus I_2, E^{\text{T}}}{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \text{exp} : \mathbf{unit} \triangleright \text{exp}', I, E^{\text{T}}_1} \text{ CHECK_EXP_NONDETBASE} \\
\\
\frac{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \mathbf{nondet} \{\text{exp}\} : \mathbf{unit} \triangleright \{\text{exp}'\}, I, E^{\text{T}}}{\langle E^{\text{T}}, E^{\text{D}} \rangle, \mathbf{unit}, \text{widening} \vdash \mathbf{nondet} \{\text{exp}\} : \mathbf{unit} \triangleright \{\text{exp}'\}, I, E^{\text{T}}} \text{ CHECK_EXP_NONDETBASE}
\end{array}$$

$$\frac{\langle E^T, E^D \rangle, \mathbf{unit}, \text{widening} \vdash \text{exp} : \mathbf{unit} \triangleright \text{exp}', I_1, E^T_1 \quad \langle (E^T \uplus E^T_1), E^D \rangle, \mathbf{unit}, \text{widening} \vdash \mathbf{nondet} \{ \overline{\text{exp}_i}^i \} : \mathbf{unit} \triangleright \{ \overline{\text{exp}'_i}^i \}, I_2, E^T_2}{\langle E^T, E^D \rangle, \mathbf{unit}, \text{widening} \vdash \mathbf{nondet} \{ \text{exp}; \overline{\text{exp}_i}^i \} : \mathbf{unit} \triangleright \{ \text{exp}'; \overline{\text{exp}'_i}^i \}, I_1 \uplus I_2, E^T} \text{CHECK_EXP_NONDETREC}$$

$$\frac{E, t, \text{widening} \vdash \text{exp} : u \triangleright \text{exp}', I_1, E^T_1 \quad E, \text{widening} \vdash \text{lexp} : t \triangleright \text{lexp}', I_2, E^T_2}{E, \mathbf{unit}, \text{widening} \vdash \text{lexp} := \text{exp} : \mathbf{unit} \triangleright \text{lexp}' := \text{exp}', I \uplus I_2, E^T_2} \text{CHECK_EXP_ASSIGN}$$

$E, \text{widening} \vdash \text{lexp} : t \triangleright \text{lexp}', I, E^T$ Check the left hand side of an assignment

$$\frac{E^T(id) \triangleright \mathbf{register} \langle t \rangle}{\langle E^T, E^D \rangle, \text{widening} \vdash id : t \triangleright id, \langle \{ \}, \{ \mathbf{wreg} \} \rangle, E^T} \text{CHECK_LEXP_WREG}$$

$$\frac{E^T(id) \triangleright \mathbf{reg} \langle t \rangle}{\langle E^T, E^D \rangle, \text{widening} \vdash id : t \triangleright id, I_\epsilon, E^T} \text{CHECK_LEXP_WLOCL}$$

$$\frac{E^T(id) \triangleright t}{\langle E^T, E^D \rangle, \text{widening} \vdash id : t \triangleright id, I_\epsilon, E^T} \text{CHECK_LEXP_VAR}$$

$$\frac{id \notin \mathbf{dom}(E^T)}{\langle E^T, E^D \rangle, \text{widening} \vdash id : t \triangleright id, I_\epsilon, \{ id \mapsto \mathbf{reg} \langle t \rangle \}} \text{CHECK_LEXP_WNEW}$$

$$\frac{E^T(id) \triangleright \mathbf{register} \langle t \rangle \quad E^D \vdash \text{typ} \rightsquigarrow u \quad E^D, \text{widening} \vdash u \lesssim t : u, \Sigma^N}{\langle E^T, E^D \rangle, \text{widening} \vdash (\text{typ})id : t \triangleright id, \langle \Sigma^N, \{ \mathbf{wreg} \} \rangle, E^T} \text{CHECK_LEXP_WREGCAST}$$

$$\frac{E^T(id) \triangleright \mathbf{reg} \langle t \rangle \quad E^D \vdash \text{typ} \rightsquigarrow u \quad E^D, \text{widening} \vdash u \lesssim t : u, \Sigma^N}{\langle E^T, E^D \rangle, \text{widening} \vdash (\text{typ})id : t \triangleright id, \langle \Sigma^N, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_WLOCLCAST}$$

$$\frac{E^T(id) \triangleright t \quad E^D \vdash \text{typ} \rightsquigarrow u \quad E^D, \text{widening} \vdash u \lesssim t : u, \Sigma^N}{\langle E^T, E^D \rangle, \text{widening} \vdash (\text{typ})id : t \triangleright id, \langle \Sigma^N, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_VARCAST}$$

$$\begin{array}{c}
\frac{id \notin \mathbf{dom}(E^T) \quad E^D \vdash typ \rightsquigarrow t}{\langle E^T, E^D \rangle, widening \vdash (typ)id : t \triangleright id, I_\epsilon, \{id \mapsto \mathbf{reg} \langle t \rangle\}} \text{CHECK_LEXP_WNEWCAST} \\
\\
\frac{\begin{array}{c} E^T(id) \triangleright E^K, \Sigma^N, \mathbf{Extern}, (t_1, \dots, t_n, t) \rightarrow t' \{ \overline{base_effect_i}^i, \mathbf{wmem}, \overline{base_effect_j}^j \} \\ \langle E^T, E^D \rangle, (t_1, \dots, t_n), widening \vdash exp : u_1 \triangleright exp', I, E^T_1 \end{array}}{\langle E^T, E^D \rangle, widening \vdash id(exp) : t \triangleright id(exp'), I \uplus \langle \Sigma^N, \{\mathbf{wmem}\} \rangle, E^T} \text{CHECK_LEXP_WMEM} \\
\\
\frac{\begin{array}{c} E^T(id) \triangleright E^K, \Sigma^N, \mathbf{Extern}, (t_1, \dots, t_n, t) \rightarrow t' \{ \overline{base_effect_i}^i, \mathbf{wreg}, \overline{base_effect_j}^j \} \\ \langle E^T, E^D \rangle, (t_1, \dots, t_n), widening \vdash exp : u_1 \triangleright exp', I, E^T_1 \end{array}}{\langle E^T, E^D \rangle, widening \vdash id(exp) : t \triangleright id(exp'), I \uplus \langle \Sigma^N, \{\mathbf{wreg}\} \rangle, E^T} \text{CHECK_LEXP_WREGCALL} \\
\\
\frac{\begin{array}{c} E, \mathbf{atom} \langle ne \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : u \triangleright exp', I_1, E^T \\ E, (\mathbf{none}, \mathbf{vectors}) \vdash lexp : \mathbf{vector} \langle ne_1 ne_2 \mathbf{inc} t \rangle \triangleright lexp', I_2, E^T \end{array}}{E, widening \vdash lexp[exp] : t \triangleright lexp'[exp'], I_1 \uplus I_2 \uplus \langle \{ne_1 \leq ne, ne_1 + ne_2 \geq ne\}, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_WBITINC} \\
\\
\frac{\begin{array}{c} E, \mathbf{atom} \langle ne \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp : u \triangleright exp', I_1, E^T \\ E, (\mathbf{none}, \mathbf{vectors}) \vdash lexp : \mathbf{vector} \langle ne_1 ne_2 \mathbf{dec} t \rangle \triangleright lexp', I_2, E^T \end{array}}{E, widening \vdash lexp[exp] : t \triangleright lexp'[exp'], I_1 \uplus I_2 \uplus \langle \{ne \leq ne_1, ne_1 + (-ne_2) \leq ne\}, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_WBITDEC} \\
\\
\frac{\begin{array}{c} E, \mathbf{atom} \langle ne_1 \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : u_1 \triangleright exp'_1, I_1, E^T \\ E, \mathbf{atom} \langle ne_2 \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_2 : u_2 \triangleright exp'_2, I_2, E^T \\ E, (\mathbf{none}, \mathbf{vectors}) \vdash lexp : \mathbf{vector} \langle ne_3 ne_4 \mathbf{inc} t \rangle \triangleright lexp', I_3, E^T \end{array}}{E, widening \vdash lexp[exp_1 : exp_2] : \mathbf{vector} \langle ne_1 ne_2 + (-ne_1) \mathbf{inc} t \rangle \triangleright lexp'[exp'_1 : exp'_2], I_1 \uplus I_2 \uplus I_3 \uplus \langle \{ne_3 \leq ne_1, ne_3 + ne_4 \leq ne_2 + (-ne_1)\}, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_WSLIC} \\
\\
\frac{\begin{array}{c} E, \mathbf{atom} \langle ne_1 \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_1 : u_1 \triangleright exp'_1, I_1, E^T \\ E, \mathbf{atom} \langle ne_2 \rangle, (\mathbf{nums}, \mathbf{none}) \vdash exp_2 : u_2 \triangleright exp'_2, I_2, E^T \\ E, (\mathbf{none}, \mathbf{vectors}) \vdash lexp : \mathbf{vector} \langle ne_3 ne_4 \mathbf{inc} t \rangle \triangleright lexp', I_3, E^T \end{array}}{E, widening \vdash lexp[exp_1 : exp_2] : \mathbf{vector} \langle ne_1 ne_2 + (-ne_1) \mathbf{inc} t \rangle \triangleright lexp'[exp'_1 : exp'_2], I_1 \uplus I_2 \uplus I_3 \uplus \langle \{ne_1 \leq ne_3, ne_3 + (-ne_4) \leq ne_1 + (-ne_2)\}, \mathbf{pure} \rangle, E^T} \text{CHECK_LEXP_WS} \\
\\
\frac{\begin{array}{c} E^R(x \langle t_args \rangle) \triangleright \overline{id_i : t_i}^i id : t \overline{id'_j : t'_j}^j \\ \langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, widening \vdash lexp : x \langle t_args \rangle \triangleright lexp', I, E^T \end{array}}{\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle, widening \vdash lexp.id : t \triangleright lexp'.id, I, E^T} \text{CHECK_LEXP_WRECORD}
\end{array}$$

$E \vdash \text{letbind} \triangleright \text{letbind}', E^T, \Sigma^N, effect, E^K$

Build the environment for a let binding, collecting index constraints

$$\begin{array}{c}
\langle E^K, E^A, E^R, E^E \rangle \vdash \text{typschm} \rightsquigarrow t, E^K_2, \Sigma^N \\
\langle E^T, \langle E^K \uplus E^K_2, E^A, E^R, E^E \rangle \rangle, t \vdash \text{pat} : u \triangleright \text{pat}', E^T_1, \Sigma^N_1 \\
\langle E^T, \langle E^K \uplus E^K_2, E^A, E^R, E^E \rangle \rangle, t, (\mathbf{none}, \mathbf{none}) \vdash \text{exp} : u' \triangleright \text{exp}', \langle \Sigma^N_2, \text{effect} \rangle, E^T_2 \\
\text{<<no parses (char 49): <E.k u+ E.k2,E.a,E.r,E.e>, (none,none) |- u' < u,*** S.N3 >>} \\
\hline
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle \vdash \mathbf{let} \text{ typschm pat} = \text{exp} \triangleright \mathbf{let} \text{ typschm pat}' = \text{exp}', E^T_1, \Sigma^N \uplus \Sigma^N_1 \uplus \Sigma^N_2 \uplus \Sigma^N_3, \text{effect}, E^K_2 & \text{CHECK_LETBIND_VAL_ANNOT} \\
\langle E^T, E^D \rangle, t \vdash \text{pat} : u \triangleright \text{pat}', E^T_1, \Sigma^N_1 \\
\text{<<no parses (char 26): <(E.t u+ E.t1),E.d>,u |- e***xp : u' gives exp',<S.N2,effect>,E.t2 >>} \\
\hline
\langle E^T, E^D \rangle \vdash \mathbf{let} \text{ pat} = \text{exp} \triangleright \mathbf{let} \text{ pat}' = \text{exp}', E^T_1, \Sigma^N_1 \uplus \Sigma^N_2, \text{effect}, \{ \} & \text{CHECK_LETBIND_VAL_NOANNOT}
\end{array}$$

$$E^D \vdash \text{type_def} \triangleright E$$

Check a type definition

$$\begin{array}{c}
E^D \vdash \text{typschm} \rightsquigarrow t, E^K, \Sigma^N \\
\hline
E^D \vdash \mathbf{typedef} \text{ id name_scm_opt} = \text{typschm} \triangleright \langle \{ \}, \langle \{ \}, \{ \text{id} \mapsto E^K, \Sigma^N, \mathbf{None}, t \}, \{ \}, \{ \} \rangle \rangle & \text{CHECK_TD_ABBREV} \\
E^D \vdash \text{typ}_1 \rightsquigarrow t_1 \quad \dots \quad E^D \vdash \text{typ}_n \rightsquigarrow t_n \\
E^R \equiv \{ \{ \text{id}_1 : t_1, \dots, \text{id}_n : t_n \} \mapsto x \} \\
\hline
E^D \vdash \mathbf{typedef} \text{ x name_scm_opt} = \mathbf{const struct} \{ \text{typ}_1 \text{ id}_1; \dots; \text{typ}_n \text{ id}_n; ? \} \triangleright \langle \{ \}, \langle \{ x \mapsto K_Typ \}, \{ \}, E^R, \{ \} \rangle \rangle & \text{CHECK_TD_UNQUANT_RECORD} \\
\langle E^K, E^A, E^R, E^E \rangle \vdash \text{quant_item}_i \rightsquigarrow E^K_i, \Sigma^N_i{}^i \\
\langle E^K \uplus \overline{E^K_i}{}^i, E^A, E^R, E^E \rangle \vdash \text{typ}_1 \rightsquigarrow t_1 \quad \dots \quad \langle E^K \uplus \overline{E^K_i}{}^i, E^A, E^R, E^E \rangle \vdash \text{typ}_n \rightsquigarrow t_n \\
\{ x'_1 \mapsto k_1, \dots, x'_m \mapsto k_m \} \equiv \uplus \overline{E^K_i}{}^i \\
E^R_1 \equiv \{ \{ \text{id}_1 : t_1, \dots, \text{id}_n : t_n \} \mapsto \{ x'_1 \mapsto k_1, \dots, x'_m \mapsto k_m \}, \uplus \overline{\Sigma^N_i}{}^i, \mathbf{None}, x \langle x'_1 \dots x'_m \rangle \} \\
E^{K'_1} \equiv \{ x \mapsto K_Lam(k_1 \dots k_m \rightarrow K_Typ) \} \\
\hline
\langle E^K, E^A, E^R, E^E \rangle \vdash \mathbf{typedef} \text{ x name_scm_opt} = \mathbf{const struct forall} \overline{\text{quant_item}_i}{}^i . \{ \text{typ}_1 \text{ id}_1; \dots; \text{typ}_n \text{ id}_n; ? \} \triangleright \langle \{ \}, \langle E^{K'}, \{ \}, E^R_1, \{ \} \rangle \rangle & \text{CHECK_TD_QUANT_RECORD} \\
E^T \equiv \{ \text{id}_1 \mapsto \{ \}, \{ \}, \mathbf{Ctor}, t_1 \rightarrow x \mathbf{pure}, \dots, \text{id}_n \mapsto \{ \}, \{ \}, \mathbf{Ctor}, t_n \rightarrow x \mathbf{pure} \} \\
E^{K_1} \equiv \{ x \mapsto K_Typ \} \\
\langle E^K \uplus E^{K_1}, E^A, E^R, E^E \rangle \vdash \text{typ}_1 \rightsquigarrow t_1 \quad \dots \quad \langle E^K \uplus E^{K_1}, E^A, E^R, E^E \rangle \vdash \text{typ}_n \rightsquigarrow t_n \\
\hline
\langle E^K, E^A, E^R, E^E \rangle \vdash \mathbf{typedef} \text{ x name_scm_opt} = \mathbf{const union} \{ \text{typ}_1 \text{ id}_1; \dots; \text{typ}_n \text{ id}_n; ? \} \triangleright \langle E^T, \langle E^{K_1}, \{ \}, \{ \}, \{ \} \rangle \rangle & \text{CHECK_TD_UNQUANT_UNION}
\end{array}$$

$$\begin{array}{c}
\overline{\langle E^K, E^A, E^R, E^E \rangle \vdash \text{quant_item}_i \rightsquigarrow E^K_i, \Sigma^N_i}^i \\
\{x'_1 \mapsto k_1, \dots, x'_m \mapsto k_m\} \equiv \uplus \overline{E^K_i}^i \\
E^{K'} \equiv \{x \mapsto K_Lam(k_1 \dots k_m \rightarrow K_Typ)\} \uplus \overline{E^K_i}^i \\
\langle E^K \uplus E^{K'}, E^A, E^R, E^E \rangle \vdash \text{typ}_1 \rightsquigarrow t_1 \quad \dots \quad \langle E^K \uplus E^{K'}, E^A, E^R, E^E \rangle \vdash \text{typ}_n \rightsquigarrow t_n \\
t \equiv x \langle x'_1 \dots x'_m \rangle \\
E^T \equiv \{id_1 \mapsto E^{K'}, \uplus \overline{\Sigma^N_i}^i, \mathbf{Ctor}, t_1 \rightarrow t \text{ pure}, \dots, id_n \mapsto E^{K'}, \uplus \overline{\Sigma^N_i}^i, \mathbf{Ctor}, t_n \rightarrow t \text{ pure}\} \\
\hline
\langle E^K, E^A, E^R, E^E \rangle \vdash \mathbf{typedef} \text{ id name_scm_opt} = \mathbf{const union forall} \overline{\text{quant_item}_i}^i . \{\text{typ}_1 id_1; \dots; \text{typ}_n id_n; ?\} \triangleright \langle E^T, \langle E^{K'}, \{\}, \{\}, \{\} \rangle \rangle
\end{array}$$

CHECK_TD_QUANT_UNION

$$\begin{array}{c}
E^T \equiv \{id_1 \mapsto x, \dots, id_n \mapsto x\} \\
E^E \equiv \{x \mapsto \{num_1 \mapsto id_1 \dots num_n \mapsto id_n\}\} \\
\hline
E^D \vdash \mathbf{typedef} \text{ x name_scm_opt} = \mathbf{enumerate} \{id_1; \dots; id_n; ?\} \triangleright \langle E^T, \langle \{id \mapsto K_Typ\}, \{\}, \{\}, E^E \rangle \rangle
\end{array}$$

CHECK_TD_ENUMERATE

$E \vdash \text{fundef} \triangleright \text{fundef}', E^T, \Sigma^N$

Check a function definition

$$\begin{array}{c}
E^T(id) \triangleright E^{K'}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect} \\
\overline{E^D \vdash \text{quant_item}_i \rightsquigarrow E^K_i, \Sigma^N_i}^i \\
\Sigma^{N''} \equiv \uplus \overline{\Sigma^N_i}^i \\
E^{K'} \equiv \overline{E^K_i}^i \\
E^D_1 \equiv \langle E^{K'}, \{\}, \{\}, \{\} \rangle \uplus E^D \\
E^D_1 \vdash \text{typ} \rightsquigarrow u \\
\text{<<no parses (char 12): E_d1 |- u <*** t, S_N2 >>} \\
\overline{\langle E^T, E^D_1 \rangle, t_1 \vdash \text{pat}_j : u_j \triangleright \text{pat}'_j, E^T_j, \Sigma^{N'''}_j}^j \\
\text{<<no parses (char 29): </<(E_t u+ E_tj),E_d1>,u |- e***xpj : u' gives expj',<S_N''''j,effect'j>,E_t'j//j/> >>} \\
\Sigma^{N''''} \equiv \Sigma^{N_2} \uplus \overline{\Sigma^{N'''}_j} \uplus \Sigma^{N''''}_j \\
\text{effect} \equiv \uplus \overline{\text{effect}'_j}^j \\
\Sigma^N \equiv \mathbf{resolve} (\Sigma^{N'} \uplus \Sigma^{N''} \uplus \Sigma^{N''''}) \\
\hline
\langle E^T, E^D \rangle \vdash \mathbf{function rec forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}_j = \text{exp}_j^j \triangleright \mathbf{function rec forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}'_j = \text{exp}'_j^j, E^T, \Sigma^N
\end{array}$$

CHECK_FD_REC

$E^T(id) \triangleright E^{K'}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}$

$E^D \vdash typ \rightsquigarrow u$

$\llcorner \text{no parses (char 11): } E_d \mid - u \llcorner \text{*** t, S_N2} \gg$

$\frac{\langle E^T, E^D \rangle, t_1 \vdash pat_j : u_j \triangleright pat', E^T_j, \Sigma^{N''_j j}}{\text{effect} \equiv \uplus \overline{effect'_j}^j}$

$\llcorner \text{no parses (char 28): } \llcorner \llcorner (E_t \text{ u+ } E_tj), E_d, u \mid - e***xpj : \text{uj' gives expj'}, \langle S_N''j, effect'j \rangle, E_t'j // j / \gg \gg$

$\text{effect} \equiv \uplus \overline{effect'_j}^j$

$\Sigma^N \equiv \mathbf{resolve}(\Sigma^{N_2} \uplus \Sigma^{N'} \uplus \overline{\Sigma^{N''_j} \uplus \Sigma^{N'''_j}}^j)$

$\langle E^T, E^D \rangle \vdash \mathbf{function\ rec\ typ\ effect\ effect\ id\ pat_j = exp_j^j} \triangleright \mathbf{function\ rec\ typ\ effect\ effect\ id\ pat'_j = exp'_j^j}, E^T, \Sigma^N$

CHECK_FD_REC_FUNCTION2

$\frac{\langle E^K, E^A, E^R, E^E \rangle \vdash quant_item_i \rightsquigarrow E^K_i, \Sigma^{N_i}{}^i}{\Sigma^{N'} \equiv \uplus \overline{\Sigma^{N_i}{}^i}}$

$\Sigma^{N'} \equiv \uplus \overline{\Sigma^{N_i}{}^i}$

$E^{K'} \equiv E^K \uplus \overline{E^K_i}^i$

$\langle E^{K'}, E^A, E^R, E^E \rangle \vdash typ \rightsquigarrow t$

$\frac{\langle E^T, \langle E^{K'}, E^A, E^R, E^E \rangle \rangle, t_1 \vdash pat_j : u_j \triangleright pat'_j, E^T_j, \Sigma^{N''_j j}}{E^{T'} \equiv (E^T \uplus \{id \mapsto E^{K'}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\})}$

$E^{T'} \equiv (E^T \uplus \{id \mapsto E^{K'}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\})$

$\llcorner \text{no parses (char 44): } \llcorner \llcorner (E_t' \text{ u+ } E_tj), \langle E_k', E_a, E_r, E_e \rangle, t \mid - e***xpj : \text{u'j gives expj'}, \langle S_N''j, effect'j \rangle, E_t'j // j / \gg \gg$

$\text{effect} \equiv \uplus \overline{effect'_j}^j$

$\Sigma^N \equiv \mathbf{resolve}(\Sigma^{N'} \uplus \overline{\Sigma^{N''_j} \uplus \Sigma^{N'''_j}}^j)$

$\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle \vdash \mathbf{function\ rec\ forall\ quant_item_i^i . typ\ effect\ effect\ id\ pat_j = exp_j^j} \triangleright \mathbf{function\ rec\ forall\ quant_item_i^i . typ\ effect\ effect\ id\ pat'_j = exp'_j^j}, E^{T'}, \Sigma^N$

C

$E^D \vdash typ \rightsquigarrow t$

$\frac{\langle E^T, E^D \rangle, t_1 \vdash pat_j : u_j \triangleright pat'_j, E^T_j, \Sigma^{N'_j j}}{E^{T'} \equiv (E^T \uplus \{id \mapsto \{\}, \{\}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\})}$

$E^{T'} \equiv (E^T \uplus \{id \mapsto \{\}, \{\}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\})$

$\llcorner \text{no parses (char 29): } \llcorner \llcorner (E_t' \text{ u+ } E_tj), E_d, t \mid - e***xpj : \text{uj' gives expj'}, \langle S_N'j, effect'j \rangle, E_t'j // j / \gg \gg$

$\text{effect} \equiv \uplus \overline{effect'_j}^j$

$\Sigma^N \equiv \mathbf{resolve}(\uplus \overline{\Sigma^{N'_j} \uplus \Sigma^{N''_j}}^j)$

$\langle E^T, E^D \rangle \vdash \mathbf{function\ rec\ typ\ effect\ effect\ id\ pat_j = exp_j^j} \triangleright \mathbf{function\ rec\ typ\ effect\ effect\ id\ pat'_j = exp'_j^j}, E^{T'}, \Sigma^N$

CHECK_FD_REC_FUNCTION_NO_SPEC

$$\begin{array}{c}
\frac{E^T(id) \triangleright E^{K'}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}}{\langle E^K, E^A, E^R, E^E \rangle \vdash \text{quant_item}_i \rightsquigarrow E^K_i, \Sigma^{N_i}_i} \\
\Sigma^{N''} \equiv \sqcup \overline{\Sigma^{N_i}_i} \\
E^{K''} \equiv \overline{E^K_i} \\
\langle E^{K''} \sqcup E^K, E^A, E^R, E^E \rangle \vdash \text{typ} \rightsquigarrow u \\
\text{<<no parses (char 35): } \langle E.k'' \text{ u+ } E.k, E.a, E.r, E.e \rangle \mid\text{- } u \text{ <*** t, S_N2 >>} \\
\frac{\langle E^T, \langle E^K \sqcup E^{K''}, E^A, E^R, E^E \rangle \rangle, t_1 \vdash \text{pat}_j : u_j \triangleright \text{pat}'_j, E^T_j, \Sigma^{N''}_j}{\text{<<no parses (char 57): } \text{</<(E.t u- id u+ E.tj), <E.k u+ E.k'', E.a, E.r, E.e>>, t} \mid\text{- e***xpj : } u_j' \text{ gives expj}', \text{<S_N''j, effect'j>, E.t'j//j/> >>} \\
\Sigma^{N''''} \equiv \sqcup \overline{\Sigma^{N''}_j \sqcup \Sigma^{N''''}_j} \\
\text{effect} \equiv \sqcup \overline{\text{effect}'_j} \\
\Sigma^N \equiv \mathbf{resolve}(\Sigma^{N'} \sqcup \Sigma^{N''} \sqcup \Sigma^{N''''}) \\
\hline
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle \vdash \mathbf{function forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}_j = \text{exp}_j^j \triangleright \mathbf{function forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}'_j = \text{exp}'_j^j, E^T, \Sigma^N \\
\frac{E^T(id) \triangleright \{ \}, \Sigma^N_1, \mathbf{Global}, t_1 \rightarrow t \text{ effect}}{E^D \vdash \text{typ} \rightsquigarrow u} \\
\text{<<no parses (char 11): } E.d \mid\text{- } u \text{ <*** t, S_N2 >>} \\
\frac{\langle E^T, E^D \rangle, t_1 \vdash \text{pat}_j : u_j \triangleright \text{pat}_j, E^T_j, \Sigma^{N'}_j}{\text{<<no parses (char 36): } \text{</<(E.t u- id u+ E.tj), E.d>, u} \mid\text{- e***xpj : } u_j' \text{ gives expj}', \text{<S_N''j, effect'j>, E.t'j//j/> >>} \\
\text{effect} \equiv \sqcup \overline{\text{effect}'_j} \\
\Sigma^N \equiv \mathbf{resolve}(\Sigma^N_1 \sqcup \Sigma^N_2 \sqcup \overline{\Sigma^{N'}_j \sqcup \Sigma^{N''}_j}) \\
\hline
\langle E^T, E^D \rangle \vdash \mathbf{function typ effect effect id pat}_j = \text{exp}_j^j \triangleright \mathbf{function typ effect effect id pat}'_j = \text{exp}'_j^j, E^T, \Sigma^N
\end{array}$$

CHECK_FD_FUNCTION2

$$\begin{array}{c}
\overline{\langle E^K, E^A, E^R, E^E \rangle \vdash \text{quant_item}_i \rightsquigarrow E^K_i, \Sigma^N_i}^i \\
\Sigma^{N'} \equiv \uplus \overline{\Sigma^N_i}^i \\
E^{K''} \equiv E^K \uplus \overline{E^K_i}^i \\
\overline{\langle E^{K''}, E^A, E^R, E^E \rangle \vdash \text{typ} \rightsquigarrow t} \\
\overline{\langle E^T, \langle E^{K''}, E^A, E^R, E^E \rangle, t_1 \vdash \text{pat}_j : u_j \triangleright \text{pat}_j, E^T_j, \Sigma^{N''j} \rangle}^j \\
E^{T'} \equiv (E^T \uplus \{id \mapsto E^{K''}, \Sigma^{N'}, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\}) \\
\text{<<no parses (char 44): } \text{</<(E_t u+ E_tj),<E_k'',E_a,E_r,E_e>>,t |- e***xpj : } \text{uj' gives expj', <S_N''j,effect'j>,E_t'j//j/> >> \\
\text{effect} \equiv \uplus \overline{\text{effect}'_j}^j \\
\Sigma^N \equiv \mathbf{resolve}(\Sigma^{N'} \uplus \overline{\Sigma^{N'}_j \uplus \Sigma^{N''j}}^j) \\
\hline
\langle E^T, \langle E^K, E^A, E^R, E^E \rangle \rangle \vdash \mathbf{function forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}_j = \text{exp}_j^j \triangleright \mathbf{function forall} \overline{\text{quant_item}_i}^i . \text{typ effect effect id pat}'_j = \text{exp}'_j^j, E^{T'}, \Sigma^N \quad \text{CHECK_L} \\
\hline
E^D \vdash \text{typ} \rightsquigarrow t \\
\overline{\langle E^T, E^D \rangle, t_1 \vdash \text{pat}_j : u_j \triangleright \text{pat}'_j, E^T_j, \Sigma^{N'}_j}^j \\
E^{T'} \equiv (E^T \uplus \{id \mapsto \{\}, \Sigma^N, \mathbf{Global}, t_1 \rightarrow t \text{ effect}\}) \\
\text{<<no parses (char 28): } \text{</<(E.t u+ E_tj),E_d>,t |- e***xpj : } \text{uj' gives exp', <S_N'j,effect'j>,E_t'j//j/> >> \\
\text{effect} \equiv \uplus \overline{\text{effect}'_j}^j \\
\Sigma^N \equiv \mathbf{resolve}(\uplus \Sigma^{N'}_j \uplus \overline{\Sigma^{N''j}}^j) \\
\hline
\langle E^T, E^D \rangle \vdash \mathbf{function typ effect effect id pat}_j = \text{exp}_j^j \triangleright \mathbf{function typ effect effect id pat}'_j = \text{exp}'_j^j, E^{T'}, \Sigma^N \quad \text{CHECK_FD_FUNCTION_NO_SPEC2} \\
\hline
\boxed{E \vdash \text{val_spec} \triangleright E^T} \quad \text{Check a value specification} \\
\hline
\overline{\langle E^T, E^D \rangle \vdash \mathbf{val} \text{typschm} id \triangleright \{id \mapsto E^K_1, \Sigma^N, \mathbf{Global}, t\}} \quad \text{CHECK_SPEC_VAL_SPEC} \\
\overline{\langle E^T, E^D \rangle \vdash \mathbf{val} \text{extern} \text{typschm} id = \text{string} \triangleright \{id \mapsto E^K_1, \Sigma^N, \mathbf{Extern}, t\}} \quad \text{CHECK_SPEC_EXTERN} \\
\hline
\boxed{E^D \vdash \text{default_spec} \triangleright E^T, E^K_1} \quad \text{Check a default typing specification} \\
\hline
\overline{\langle E^K, E^A, E^R, E^E \rangle \vdash \mathbf{default} \text{base_kind}'x \triangleright \{\}, \{x \mapsto k \mathbf{default}\}} \quad \text{CHECK_DEFAULT_KIND}
\end{array}$$

$$\frac{E^D \vdash \text{typschm} \rightsquigarrow t, E^K_1, \Sigma^N}{E^D \vdash \mathbf{default} \text{ typschm } id \triangleright \{id \mapsto E^K_1, \Sigma^N, \mathbf{Default}, t\}, \{\}} \quad \text{CHECK_DEFAULT_TYP}$$

$$\boxed{E \vdash \text{def} \triangleright \text{def}', E'}$$

Check a definition

$$\frac{E^D \vdash \text{type_def} \triangleright E}{\langle E^T, E^D \rangle \vdash \text{type_def} \triangleright \text{type_def}, \langle E^T, E^D \rangle \uplus E} \quad \text{CHECK_DEF_TDEF}$$

$$\frac{E \vdash \text{fundef} \triangleright \text{fundef}', E^T, \Sigma^N}{E \vdash \text{fundef} \triangleright \text{fundef}', E \uplus \langle E^T, \epsilon \rangle} \quad \text{CHECK_DEF_FDEF}$$

$$\frac{\begin{array}{l} E \vdash \text{letbind} \triangleright \text{letbind}', \{id_1 \mapsto t_1, \dots, id_n \mapsto t_n\}, \Sigma^N, \mathbf{pure}, E^K \\ \Sigma^N_1 \equiv \mathbf{resolve}(\Sigma^N) \end{array}}{E \vdash \text{letbind} \triangleright \text{letbind}', E \uplus \langle \{id_1 \mapsto E^K, \Sigma^N, \mathbf{None}, t_1, \dots, id_n \mapsto E^K, \Sigma^N, \mathbf{None}, t_n\}, \epsilon \rangle} \quad \text{CHECK_DEF_VDEF}$$

$$\frac{E \vdash \text{val_spec} \triangleright E^T}{E \vdash \text{val_spec} \triangleright \text{val_spec}, E \uplus \langle E^T, \epsilon \rangle} \quad \text{CHECK_DEF_VSPEC}$$

$$\frac{E^D \vdash \text{default_spec} \triangleright E^T_1, E^K_1}{\langle E^T, E^D \rangle \vdash \text{default_spec} \triangleright \text{default_spec}, \langle (E^T \uplus E^T_1), E^D \uplus \langle E^K_1, \{\}, \{\}, \{\} \rangle \rangle} \quad \text{CHECK_DEF_DEFAULT}$$

$$\frac{E^D \vdash \text{typ} \rightsquigarrow t}{\langle E^T, E^D \rangle \vdash \mathbf{register} \text{ typ } id \triangleright \mathbf{register} \text{ typ } id, \langle (E^T \uplus \{id \mapsto \mathbf{register} \langle t \rangle\}), E^D \rangle} \quad \text{CHECK_DEF_REGISTER}$$

$$\boxed{E \vdash \text{defs} \triangleright \text{defs}', E'}$$

Check definitions, potentially given default environment of built-in library

$$\frac{\begin{array}{l} E \vdash \text{def} \triangleright \text{def}', E_1 \\ E \uplus E_1 \vdash \overline{\text{def}_i}^i \triangleright \overline{\text{def}'_i}^i, E_2 \end{array}}{E \vdash \text{def} \overline{\text{def}_i}^i \triangleright \text{def}' \overline{\text{def}'_i}^i, E_2} \quad \text{CHECK_DEFS_DEFS}$$

6 Sail operational semantics {TODO}