c	numeric value
a	array
x	term variable
f	function name
n	index variable
k	index variable
i	index variable
b_h	boolean value
c_h	numeric value
f_h	function name
$rval$	"return value" variable
$rnset$	"rval-not-set" variable

v	$::=$		values
	$\mid$	$\textsc{True}$	bitmask true ($\texttt{0b1111...}$)
	$\mid$	$\textsc{False}$	bitmask false ($\texttt{0b0000...}$)
	$\mid$	c	numeric value
	$\mid$	a	bytearray
$\ominus$	$::=$		unary operations
	$\mid$	$\texttt{\~{}}$	bitwise not
$\oplus$	$::=$		binary operations
	$\mid$	$\texttt{+}$	
	$\mid$	$\texttt{-}$	
	$\mid$	$\texttt{*}$	
	$\mid$	$\texttt{<<}$	
	$\mid$	$\texttt{>>}$	
	$\mid$	$\texttt{\&}$	bitwise and
	$\mid$	$\texttt{\vert}$	bitwise or
	$\mid$	$\texttt{==}_{\texttt{s}}$	equals (sign extended)
	$\mid$	$\texttt{!=}_{\texttt{s}}$	
	$\mid$	$\texttt{>}_{\texttt{s}}$	
	$\mid$	$\texttt{<}_{\texttt{s}}$	
	$\mid$	$\texttt{>=}_{\texttt{s}}$	
	$\mid$	$\texttt{<=}_{\texttt{s}}$	
$\{\sigma\}$	$::=$		variable substitution
	$\mid$	$\{x_1/v_1, .., x_k/v_k\}$	
fval	$::=$		function spec
	$\mid$	$(x_1, .., x_n) : s \ \texttt{@}e$	
fndef	$::=$		function definition
	$\mid$	$\mathbf{fdef}\ f\ \mathit{fval}$	
$\mathit{program}$	$::=$		program
	$\mid$	$\mathit{fndef}_1; ..; \mathit{fndef}_n;\ \mathbf{expose}\ \mathit{fndef}$	list of fdefs
Λ	$::=$		function store
	$\mid$	$\emptyset_\Lambda$	empty function store
	$\mid$	$\Lambda[f \mapsto \mathit{fval}]$	define function
Γ	$::=$		global memory
	$\mid$	$\emptyset_\Gamma$	
	$\mid$	$\Gamma[a \mapsto \texttt{[]}]$	new array
	$\mid$	$\Gamma[a \mapsto \Gamma(a)[v_1 \mapsto v_2]]$	array update
μ	$::=$		local memory
	$\mid$	$\emptyset_\mu$	empty memory

	$\mid$	$\mu[x \mapsto v]$	add/update variable
	$\mid$	$\mu_1 \triangleright \mu_2$	push stack frame
κ	$::=$		program transcript
	$\mid$	$\emptyset_\kappa$	empty transcript
	$\mid$	$\kappa \triangleright \ominus$	add $\ominus$ to transcript
	$\mid$	$\kappa \triangleright \oplus$	add $\oplus$ to transcript
	$\mid$	$\kappa \triangleright$ **load**	add memory load to transcript
	$\mid$	$\kappa \triangleright$ **store**	add memory store to transcript
	$\mid$	$\kappa \triangleright f$	add call to f to transcript
	$\mid$	$\kappa \triangleright$ **ret**	add function return to transcript
e	$::=$		expressions
	$\mid$	TRUE	bitmask true (`0b1111...`)
	$\mid$	FALSE	bitmask false (`0b0000...`)
	$\mid$	c	numeric value
	$\mid$	a	bytearray
	$\mid$	x	variable
	$\mid$	$a[e]$	array access
	$\mid$	$\ominus e$	unary operation
	$\mid$	$e_1 \oplus e_2$	binary operation
	$\mid$	$f(e_1, .., e_n)$	function application
s	$::=$		statements
	$\mid$	**skip**	skip
	$\mid$	$s_1; s_2$	sequence
	$\mid$	**def** $x := e$	variable declaration
	$\mid$	**adef** $x := a$	array declaration
	$\mid$	$x := e$	variable assignment
	$\mid$	$a[e_1] := e_2$	array assignment
	$\mid$	**for** x **from** v_1 **to** $v_2 : s$	for loop
v_h	$::=$		values
	$\mid$	b_h	boolean value
	$\mid$	c_h	numeric value
	$\mid$	a	bytearray
$\ominus_h$	$::=$		unary operations
	$\mid$	`!`	logical not
	$\mid$	`~`	bitwise not
$\oplus_h$	$::=$		binary operations
	$\mid$	`+`	
	$\mid$	`-`	
	$\mid$	`*`	
	$\mid$	`<<`	
	$\mid$	`>>`	

	$\vert$	$\texttt{\&}$	
	$\vert$	$\texttt{\vert}$	
	$\vert$	$\texttt{\&\&}$	
	$\vert$	$\texttt{\vert\vert}$	
	$\vert$	$\texttt{==}$	
	$\vert$	$\texttt{!=}$	
	$\vert$	$\texttt{>}$	
	$\vert$	$\texttt{<}$	
	$\vert$	$\texttt{>=}$	
	$\vert$	$\texttt{<=}$	
e_h	$::=$		expressions
	$\vert$	b_h	boolean value
	$\vert$	c_h	numeric value
	$\vert$	a	bytearray
	$\vert$	x	variable
	$\vert$	$a[e_h]$	array access
	$\vert$	$\ominus_h e_h$	unary operation
	$\vert$	$e_{h1} \oplus e_{h2}$	binary operation
	$\vert$	$f_h(e_{h1}, .., e_{hn})$	function application
s_h	$::=$		statements
	$\vert$	$\mathbf{skip}_h$	skip
	$\vert$	$s_{h1}; s_{h2}$	sequence
	$\vert$	$\mathbf{def}_h\ x := e_h$	variable declaration
	$\vert$	$\mathbf{adef}_h\ x := a$	array declaration
	$\vert$	$x := e_h$	variable assignment
	$\vert$	$a[e_{h1}] := e_{h2}$	array assignment
	$\vert$	$\mathbf{for}_h\ x\ \mathbf{from}\ v_{h1}\ \mathbf{to}\ v_{h2}$	for loop
	$\vert$	$\mathbf{if}_h\ e_h\ \mathbf{then}\ s_{h1}\ \mathbf{else}\ s_{h2}$	conditional branch
	$\vert$	$\mathbf{return}_h\ e_h$	return
$hfval$	$::=$		function spec
	$\vert$	$(x_1, .., x_n) : s_h$	
$hfndef$	$::=$		function definition
	$\vert$	$\mathbf{fdef}_h\ f_h\ hfval$	
$hprogram$	$::=$		program
	$\vert$	$hfndef_1; ..; hfndef_n;\ \mathbf{expose}\ hfndef$	list of fdefs
ctx	$::=$		branch context
	$\vert$	x	variable
	$\vert$	$\ominus ctx$	unary operation
	$\vert$	$ctx_1 \oplus ctx_2$	binary operation

$\boxed{\{\Lambda, \Gamma, \mu, \kappa\}\ e \longrightarrow \{\Lambda', \Gamma', \mu', \kappa'\}\ e'}$ e reduces to e'

$$\frac{\begin{array}{c}\mu = \mu'[x \mapsto v] \\ \kappa' = \kappa \triangleright \mathbf{load}\end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ x \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v} \quad \textsc{Exr\_var}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e'}{\{\Lambda,\Gamma,\mu,\kappa\}\ a[e] \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ a[e']}\ \textsc{Exr\_arr\_get\_expr}$$

$$\frac{\begin{array}{l} v' = \Gamma(a)[v] \\ \kappa' = \kappa \triangleright \mathbf{load} \end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\ a[v] \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ v'}\ \textsc{Exr\_arr\_get\_val}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e'}{\{\Lambda,\Gamma,\mu,\kappa\}\ \ominus e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ \ominus e'}\ \textsc{Exr\_unop\_expr}$$

$$\frac{\begin{array}{l} v' \equiv [\![\ominus v]\!] \\ \kappa' = \kappa \triangleright \ominus \end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\ \ominus v \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ v'}\ \textsc{Exr\_unop\_val}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e_1 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e_1'}{\{\Lambda,\Gamma,\mu,\kappa\}\ e_1 \oplus e_2 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e_1' \oplus e_2}\ \textsc{Exr\_binop\_l}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e_2 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e_2'}{\{\Lambda,\Gamma,\mu,\kappa\}\ v \oplus e_2 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ v \oplus e_2'}\ \textsc{Exr\_binop\_r}$$

$$\frac{\begin{array}{l} v_3 \equiv [\![v_1 \oplus v_2]\!] \\ \kappa' = \kappa \triangleright \oplus \end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\ v_1 \oplus v_2 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ v_3}\ \textsc{Exr\_binop\_val}$$

$$\frac{}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\,\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa\}\ e}\ \textsc{Exr\_subst\_empty}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e'}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ \{\sigma\}\ e'}\ \textsc{Exr\_subst\_expr}$$

$$\frac{\kappa' = \kappa \triangleright \mathbf{load}}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{x_1/v_1,\ \ldots, x_k/v_k\}\ x_i \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ v_i}\ \textsc{Exr\_subst\_var}$$

$$\frac{}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ x \longrightarrow \{\Lambda,\Gamma,\mu,\kappa\}\ x}\ \textsc{Exr\_subst\_no\_var}$$

$$\frac{}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ v \longrightarrow \{\Lambda,\Gamma,\mu,\kappa\}\ v}\ \textsc{Exr\_subst\_val}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ e_1 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e_1'}{\{\Lambda,\Gamma,\mu,\kappa\}\, f(v_1,\ ..\ , v_k, e_1, e_2,\ ..\ , e_n) \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\, f(v_1,\ ..\ , v_k, e_1', e_2,\ ..\ , e_n)}\ \textsc{Exr\_fn\_expr}$$

$$\frac{\kappa' = \kappa \triangleright \mathbf{store}}{\{\Lambda,\Gamma,\mu,\kappa\}\, f(x_1'/v_1',\ ..\ , x_k'/v_k', v_1, v_2,\ ..\ , v_n) \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\, f(x_1'/v_1',\ ..\ , x_k'/v_k', x_1/v_1, v_2,\ ..\ , v_n)}\ \textsc{Exr\_fn\_subst}$$

$$\frac{\begin{array}{l} \Lambda = \Lambda'[f \mapsto (x_1,\ ..\ , x_k) : s\ @e] \\ \mu' = \mu \triangleright \emptyset_\mu \\ \kappa' = \kappa \triangleright f \end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\, f(x_1/v_1,\ ..\ , x_k/v_k) \longrightarrow \{\Lambda,\Gamma,\mu',\kappa'\}\ \{x_1/v_1,\ ..\ , x_k/v_k\}\ s\ @e}\ \textsc{Exr\_fn\_call}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\ e'}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\, \mathbf{skip}\ @e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\, \mathbf{skip}\ @e'}\ \textsc{Exr\_skip\_expr}$$

$$\frac{\begin{array}{l} \mu = \mu_1 \triangleright \mu_2 \\ \kappa' = \kappa \triangleright \mathbf{ret} \end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\, \mathbf{skip}\ @v \longrightarrow \{\Lambda,\Gamma,\mu_1,\kappa'\}\ v}\ \textsc{Exr\_skip\_val}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ s_1\ @e_0 \longrightarrow \{\Lambda,\Gamma,\mu',\kappa'\}\ \{\sigma'\}\ s_1'\ @e_0}{\{\Lambda,\Gamma,\mu,\kappa\}\ \{\sigma\}\ s_1; s_2\ @e_0 \longrightarrow \{\Lambda,\Gamma,\mu',\kappa'\}\ \{\sigma'\}\ s_1'; s_2\ @e_0}\ \textsc{Exr\_seq}$$

$$\frac{}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{skip};\,s\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,s\ \texttt{@}e_0}\ \textsc{Exr\_seq\_skip}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,e'}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{def}\ x := e\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma\}\,\textbf{def}\ x := e'\ \texttt{@}e_0}\ \textsc{Exr\_def\_expr}$$

$$\frac{\begin{array}{c}\mu' = \mu[x \mapsto v]\\ \kappa' = \kappa \triangleright \textbf{store}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{def}\ x := v\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu',\kappa'\}\,\{\sigma\}\,\textbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_def\_val}$$

$$\frac{\begin{array}{c}\Gamma' = \Gamma[a \mapsto \texttt{[]}]\\ \mu' = \mu[x \mapsto a]\\ \kappa' = \kappa \triangleright \textbf{store}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{adef}\ x := a\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma',\mu',\kappa'\}\,\{\sigma\}\,\textbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_def\_arr}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,e \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,e'}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,x := e\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma\}\,x := e'\ \texttt{@}e_0}\ \textsc{Exr\_assign\_expr}$$

$$\frac{\begin{array}{c}\mu' = \mu[x \mapsto v]\\ \kappa' = \kappa \triangleright \textbf{store}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,x := v\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu',\kappa'\}\,\{\sigma\}\,\textbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_assign\_val}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,e_1 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,e_1'}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,a[e_1] := e_2\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma\}\,a[e_1'] := e_2\ \texttt{@}e_0}\ \textsc{Exr\_arr\_assign\_expr\_l}$$

$$\frac{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,e_2 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,e_2'}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,a[v_1] := e_2\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma\}\,a[v_1] := e_2'\ \texttt{@}e_0}\ \textsc{Exr\_arr\_assign\_expr\_r}$$

$$\frac{\begin{array}{c}\Gamma' = \Gamma[a \mapsto \Gamma(a)[v_1 \mapsto v_2]]\\ \kappa' = \kappa \triangleright \textbf{store}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,a[v_1] := v_2\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma',\mu,\kappa'\}\,\{\sigma\}\,\textbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_arr\_assign\_val}$$

$$\frac{\begin{array}{c}v_1 < v_2\\ v_1' = v_1 + 1\\ \kappa' = \kappa \triangleright \textbf{store}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{for}\ x\ \textbf{from}\ v_1\ \textbf{to}\ v_2 : s\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma\}\,(\{x/v_1\}\,s);\textbf{for}\ x\ \textbf{from}\ v_1'\ \textbf{to}\ v_2 : s\ \texttt{@}e_0}\ \textsc{Exr\_for}$$

$$\frac{\begin{array}{c}\{\sigma_1\} \cap \{\sigma_2\} = \{\,\}\\ \{\sigma_3\} = \{\sigma_1\} \cup \{\sigma_2\}\end{array}}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma_1\}\,(\{\sigma_2\}\,s)\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu,\kappa'\}\,\{\sigma_3\}\,s\ \texttt{@}e_0}\ \textsc{Exr\_add\_subst}$$

$$\frac{v_1 = v_2}{\{\Lambda,\Gamma,\mu,\kappa\}\,\{\sigma\}\,\textbf{for}\ x\ \textbf{from}\ v_2\ \textbf{to}\ v_2 : s\ \texttt{@}e_0 \longrightarrow \{\Lambda,\Gamma,\mu',\kappa\}\,\{\sigma\}\,\textbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_for\_base}$$

$\boxed{[\![e_h]\!]_t = e}$ e_h is transformed to e

$$\frac{v \equiv [\![v_h]\!]_{int}}{[\![v_h]\!]_t = v}\ \textsc{Ext\_val}$$

$$\frac{}{[\![x]\!]_t = x}\ \textsc{Ext\_var}$$

$$\frac{}{[\![a]\!]_t = a}\ \textsc{Ext\_arr}$$

$$\frac{[\![e_h]\!]_t = e}{[\![a[e_h]]\!]_t = a[e]}\ \textsc{Ext\_arr\_get}$$

$$\frac{[\![\mathbf{fdef}_h\ f_h\ \mathit{hfval}]\!]_t = \mathbf{fdef}\, f\, \mathit{fval} \quad [\![e_{h1}]\!]_t = e_1 \quad .. \quad [\![e_{hk}]\!]_t = e_k}{[\![f_h(e_{h1}, .., e_{hk})]\!]_t = f(e_1, .., e_k)} \quad \textsc{Ext\_fn\_call}$$

$\boxed{[\![s_h]\!]_{ctx} = s}$ s_h is transformed to s

$$\frac{}{[\![\mathbf{skip}_h]\!]_{ctx} = \mathbf{skip}} \quad \textsc{Stt\_skip}$$

$$\frac{[\![s_{h1}]\!]_{ctx} = s_1 \quad [\![s_{h2}]\!]_{ctx} = s_2}{[\![s_{h1}; s_{h2}]\!]_{ctx} = s_1; s_2} \quad \textsc{Stt\_seq}$$

$$\frac{[\![e_h]\!]_t = e}{[\![\mathbf{def}_h\ x := e_h]\!]_{ctx} = \mathbf{def}\ x := e} \quad \textsc{Stt\_var\_dec}$$

$$\frac{}{[\![\mathbf{adef}_h\ x := a]\!]_{ctx} = \mathbf{adef}\ x := a} \quad \textsc{Stt\_arr\_dec}$$

$$\frac{[\![e_h]\!]_t = e \quad e' = ctx \,\&\, rnset \quad e'' = e \,\&\, e' \quad e''' = x \,\&\, ({\sim}e')}{[\![x := e_h]\!]_{ctx} = x := (e'' \mid e''')} \quad \textsc{Stt\_var\_assign}$$

$$\frac{[\![e_{h1}]\!]_t = e_1 \quad [\![e_{h2}]\!]_t = e_2 \quad e' = ctx \,\&\, rnset \quad e'' = e_2 \,\&\, e' \quad e''' = a[e_1] \,\&\, ({\sim}e')}{[\![a[e_{h1}] := e_{h2}]\!]_{ctx} = a[e_1] := (e'' \mid e''')} \quad \textsc{Stt\_arr\_assign}$$

$$\frac{[\![v_{h1}]\!]_t = v_1 \quad [\![v_{h2}]\!]_t = v_2 \quad [\![s_h]\!]_{ctx} = s}{[\![\mathbf{for}_h\ x\ \mathbf{from}\ v_{h1}\ \mathbf{to}\ v_{h2}]\!]_{ctx} = \mathbf{for}\, x\, \mathbf{from}\, v_1\, \mathbf{to}\, v_2 : s} \quad \textsc{Stt\_for}$$

$$\frac{[\![e_h]\!]_t = e \quad [\![s_{h1}]\!]_{(ctx' \,\&\, ctx)} = s_1 \quad [\![s_{h2}]\!]_{(ctx' \,\&\, ctx)} = s_2}{[\![\mathbf{if}_h\ e_h\ \mathbf{then}\ s_{h1}\ \mathbf{else}\ s_{h2}]\!]_{ctx} = \mathbf{def}\ ctx' := e; s_1; ctx' := ({\sim}ctx'); s_2} \quad \textsc{Stt\_if}$$

$$\frac{[\![e_h]\!]_t = e \quad e' = e \,\&\, (ctx \,\&\, rnset)}{[\![\mathbf{return}_h\ e_h]\!]_{ctx} = rval := (e' \mid rval); rnset := (rnset \,\&\, ({\sim}ctx))} \quad \textsc{Stt\_ret}$$

$\boxed{[\![\mathit{hfndef}]\!]_t = \mathit{fndef}}$ hfndef is transformed to fndef

$$\frac{[\![s_h]\!]_{\textsc{True}} = s}{[\![\mathbf{fdef}_h\ f_h\ (x_1, .., x_k) : s_h]\!]_t = \mathbf{fdef}\, f\, (x_1, .., x_k) : \mathbf{def}\ rval := \textsc{False}; \mathbf{def}\ rnset := \textsc{True}; s\ \texttt{@}rval} \quad \textsc{Fdeft\_fdef}$$

$\boxed{[\![\ominus_h]\!]_t = \ominus}$ $\ominus_h$ is transformed to $\ominus$

$$\frac{}{[\![\,!\,]\!]_t = {\sim}} \quad \textsc{Unopt\_lnot}$$

$$\frac{}{[\![\ominus_h]\!]_t = \ominus} \quad \textsc{Unopt\_unop}$$

$\boxed{[\![\oplus_h]\!]_t = \oplus}$ $\quad \oplus_h$ is transformed to $\oplus$

$$\frac{}{[\![\texttt{\&\&}]\!]_t = \texttt{\&}} \quad \textsc{Binopt\_land}$$

$$\frac{}{[\![\texttt{||}]\!]_t = \texttt{|}} \quad \textsc{Binopt\_lor}$$

$$\frac{}{[\![\texttt{==}]\!]_t = \texttt{==}_{\mathsf{s}}} \quad \textsc{Binopt\_eq}$$

$$\frac{}{[\![\texttt{!=}]\!]_t = \texttt{!=}_{\mathsf{s}}} \quad \textsc{Binopt\_neq}$$

$$\frac{}{[\![\texttt{>}]\!]_t = \texttt{>}_{\mathsf{s}}} \quad \textsc{Binopt\_gt}$$

$$\frac{}{[\![\texttt{<}]\!]_t = \texttt{<}_{\mathsf{s}}} \quad \textsc{Binopt\_lt}$$

$$\frac{}{[\![\texttt{>=}]\!]_t = \texttt{>=}_{\mathsf{s}}} \quad \textsc{Binopt\_gte}$$

$$\frac{}{[\![\texttt{<=}]\!]_t = \texttt{<=}_{\mathsf{s}}} \quad \textsc{Binopt\_lte}$$

$$\frac{}{[\![\oplus_h]\!]_t = \oplus} \quad \textsc{Binopt\_binop}$$

```
Definition rules:        57 good    0 bad
Definition rule clauses: 124 good    0 bad
```