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

$$\begin{array}{llll}
v & ::= & & \text{values} \\
 & | & \textsc{True} & \quad \text{bitmask true } (\texttt{0b1111...}) \\
 & | & \textsc{False} & \quad \text{bitmask false } (\texttt{0b0000...}) \\
 & | & c & \quad \text{numeric value} \\
 & | & a & \quad \text{bytearray}
\end{array}$$

$$\begin{array}{llll}
\ominus & ::= & & \text{unary operations} \\
 & | & \texttt{\textasciitilde} & \quad \text{bitwise not}
\end{array}$$

$$\begin{array}{llll}
\oplus & ::= & & \text{binary operations} \\
 & | & \texttt{+} & \\
 & | & \texttt{-} & \\
 & | & \texttt{*} & \\
 & | & \texttt{<<} & \\
 & | & \texttt{>>} & \\
 & | & \texttt{\&} & \quad \text{bitwise and} \\
 & | & \texttt{|} & \quad \text{bitwise or} \\
 & | & \texttt{==}_{\texttt{s}} & \quad \text{equals (sign extended)} \\
 & | & \texttt{!=}_{\texttt{s}} & \\
 & | & \texttt{>}_{\texttt{s}} & \\
 & | & \texttt{<}_{\texttt{s}} & \\
 & | & \texttt{>=}_{\texttt{s}} & \\
 & | & \texttt{<=}_{\texttt{s}} &
\end{array}$$

$$\begin{array}{llll}
\{\sigma\} & ::= & & \text{variable substitution} \\
 & | & \{x_1/v_1, \, .., x_k/v_k\} &
\end{array}$$

$$\begin{array}{llll}
\mathit{fval} & ::= & & \text{function spec} \\
 & | & (x_1, \, .., x_n) : s \ \texttt{@}e &
\end{array}$$

$$\begin{array}{llll}
\mathit{fndef} & ::= & & \text{function definition} \\
 & | & \mathbf{fdef}\, f\ \mathit{fval} &
\end{array}$$

$$\begin{array}{llll}
\mathit{program} & ::= & & \text{program} \\
 & | & \mathit{fndef}_1; \, ..; \mathit{fndef}_n; \ \mathbf{expose}\, \mathit{fndef} & \quad \text{list of fdefs}
\end{array}$$

$$\begin{array}{llll}
\Lambda & ::= & & \text{function store} \\
 & | & \emptyset_\Lambda & \quad \text{empty function store} \\
 & | & \Lambda[f \mapsto \mathit{fval}] & \quad \text{define function}
\end{array}$$

$$\begin{array}{llll}
\Gamma & ::= & & \text{global memory} \\
 & | & \emptyset_\Gamma & \\
 & | & \Gamma[a \mapsto \texttt{[]}] & \quad \text{new array} \\
 & | & \Gamma[a \mapsto \Gamma(a)[v_1 \mapsto v_2]] & \quad \text{array update}
\end{array}$$

$$\begin{array}{llll}
\mu & ::= & & \text{local memory} \\
 & | & \emptyset_\mu & \quad \text{empty memory}
\end{array}$$

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

$|$ `&`
$|$ `|`
$|$ `&&`
$|$ `||`
$|$ `==`
$|$ `!=`
$|$ `>`
$|$ `<`
$|$ `>=`
$|$ `<=`

e_h ::= — expressions
$|$ b_h — boolean value
$|$ c_h — numeric value
$|$ a — bytearray
$|$ x — variable
$|$ $a[e_h]$ — array access
$|$ $\ominus_h e_h$ — unary operation
$|$ $e_{h1} \oplus e_{h2}$ — binary operation
$|$ $f_h(e_{h1}, .., e_{hn})$ — function application

s_h ::= — statements
$|$ $\mathbf{skip}_h$ — skip
$|$ $s_{h1}; s_{h2}$ — sequence
$|$ $\mathbf{def}_h\ x := e_h$ — variable declaration
$|$ $\mathbf{adef}_h\ x := a$ — array declaration
$|$ $x := e_h$ — variable assignment
$|$ $a[e_{h1}] := e_{h2}$ — array assignment
$|$ $\mathbf{for}_h\ x\ \mathbf{from}\ v_{h1}\ \mathbf{to}\ v_{h2}$ — for loop
$|$ $\mathbf{if}_h\ e_h\ \mathbf{then}\ s_{h1}\ \mathbf{else}\ s_{h2}$ — conditional branch
$|$ $\mathbf{return}_h\ e_h$ — return

hfval ::= — function spec
$|$ $(x_1, .., x_n) : s_h$

hfndef ::= — function definition
$|$ $\mathbf{fdef}_h\ f_h\ \mathit{hfval}$

$\mathit{hprogram}$ ::= — program
$|$ $\mathit{hfndef}_1; ..; \mathit{hfndef}_n; \mathbf{expose}\, \mathit{hfndef}$ — list of fdefs

ctx ::= — branch context
$|$ x — variable
$|$ $\ominus \mathit{ctx}$ — unary operation
$|$ $\mathit{ctx}_1 \oplus \mathit{ctx}_2$ — binary operation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boxed{[\![\oplus_h]\!]_t = \oplus}$ $\oplus_h$ 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: 125 good    0 bad
```